IRE Transactions on ELECTRONIC COMPUTERS

Volume EC-10 **December, 1961** **Number 4**

Published Quarterly

TABLE OF CONTENTS

PUBLISHED BY THE

Professional Group on ELECTRONIC COMPUTERS

IRE PROFESSIONAL GROUP ON ELECTRONIC COMPUTERS

The Professional Group on Electronic Computers is an association of IRE members with professional interest in the field of Electronic Computers. All IRE members are eligible for membership, and will receive all Group publications upon payment of a fee of $4.00 per year, 1961. Members of certain other professional societies are eligible to be Affiliates of PGEC. See the inside back cover.

PGEC OFFICERS

ARNOLD A. COHEN, *Chairman*
Remington Rand Univac, St. Paul 16, Minn.

WALTER L. ANDERSON, *Vice Chairman*
General Kinetics, Inc., 2611 Shirlington Rd.
Arlington 6, Va.

EDWARD D. ZIMMER, *Secretary-Treasurer*
Control Data Corp., 501 Park Ave.
Minneapolis 15, Minn.

PGEC ADMINISTRATIVE COMMITTEE

Term Ending 1962
W. L. ANDERSON
R. D. ENDRES
Y. M. HILL
E. C. JOHNSON
L. G. F. JONES
R. W. MELVILLE
M. J. RELIS

Term Ending 1963
W. E. BRADLEY
W. T. CLARY
A. A. COHEN
L. FEIN
S. LUBKIN
R. L. SISSON
L. J. SPIEKER

Term Ending 1964
J. D. KENNEDY
C. SANKEY
R. M. STEWART, JR.

COMMITTEES

AFIPS Directors for IRE
W. BUCHHOLZ
A. A. COHEN
F. E. HEART
H. T. LARSON

Awards
W. L. ANDERSON, *Chairman*

Bibliography
L. F. G. JONES, *Chairman*

Chapter Activities
HENRY S. FORREST, *Chairman*

Conferences
E. C. JOHNSON, JR., *Chairman*

Constitution and Bylaws
R. L. SISSON, *Chairman*

Fellows
E. R. PIORE, *Chairman*

Membership
J. D. KENNEDY, *Chairman*

Publications Advisory
W. BUCHHOLZ, *Chairman*

Student Activities
R. M. STEWART, JR., *Chairman*

EDITORIAL BOARD

NORMAN R. SCOTT, *Editor-in-Chief*
JOHN E. SHERMAN, *Associate Editor for Analog and Hybrid Computers*
EDWARD J. MCCLUSKEY, JR., *Associate Editor for Logic and Switching Theory*
THOMAS C. BARTEE, *Reviews Editor*

IRE Transactions® on Electronic Computers

Published by The Institute of Radio Engineers, Inc., for the Professional Group on Electronic Computers at 1 East 79 Street, New York 21, N.Y. Responsibility for the contents rests upon the authors and not upon the IRE, the Group, or its members. Individual copies of this issue and all available back issues may be purchased at the following prices: IRE members (one copy) $2.25, libraries and colleges $3.25, all others $4.50. Annual subscription price: libraries and colleges $12.75; non-members $17.00. Address requests to the Institute of Radio Engineers, 1 East 79 Street, N.Y. 21, N.Y.

Notice to Authors: Address all papers and editorial correspondence to the appropriate Editor. Addresses are listed on page 343. To avoid delay and inconvenience in the processing of manuscripts, please follow the procedure suggested there.

COPYRIGHT © 1962—THE INSTITUTE OF RADIO ENGINEERS, INC.
Printed in U.S.A.

All rights, including translation, are reserved by the IRE. Requests for republication privileges should be addressed to the Institute of Radio Engineers.

The Cascade Decomposition of Sequential Machines*

M. YOELI†

Summary—This paper studies composite sequential machines obtained from smaller component machines by their connection in cascade, that is, the outputs from one component are the inputs to the next.

Given the specification of a deterministic, completely specified, synchronous, sequential machine (Mealy model), a criterion is derived for such a specification to be decomposable into specifications of smaller machines, the cascading of which will lead to a realization of the original machine required.

A simple technique, based on homomorphisms between directed graphs, is arrived at for the actual breaking up of a decomposable specification. A number of additional problems related to sequential machine decompositions are pointed out as concluding remarks.

I. Introduction

THIS paper is concerned with cascaded sequential networks (see Fig. 1). It investigates the properties of such networks and the inverse problem of decomposition, *i.e.*, the representation of a specified sequential machine as a cascaded network.

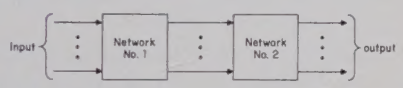


Fig. 1—Cascaded networks.

The importance of a theoretical investigation on possible decompositions of sequential machines has already been emphasized by Moore.[1] The engineering advantages of breaking down the over-all specification of a large machine into several small machines (with the number of states of the large machine being the product of the numbers of states of the small machines) are rather obvious. However, the exact evaluation of any such decomposition will depend on the particular application and cannot easily be generalized. The decomposition of information processing machines into component machines connected in parallel has been studied by Hartmanis,[2] who applied the algebraic theory of congruence relations, of Birkhoff[3] and Krishnan.[4]

Fig. 2 indicates a suitable interpretation of the concept of parallel composition of information networks. Both the parallel and cascaded compositions may be considered as extreme cases of *partially cascaded* networks, as shown in Fig. 3. On the other hand, partial cascading may obviously be viewed as a special case of (complete) cascading with some of the input leads connected straight through in both component networks. In some cases, a decomposition of a machine into two partially cascaded networks will be required with the interconnections between the component machines restricted to a minimum. The (complete) cascade decomposition studied in this paper may then serve as the initial (and essential) step toward the desired goal.

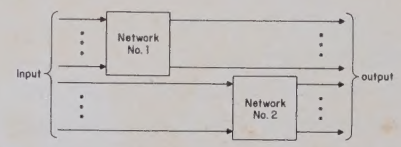


Fig. 2—Parallel composition of networks.

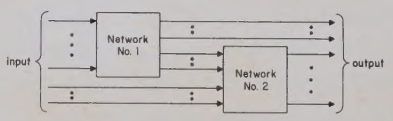


Fig. 3—Partially cascaded networks.

The paper arrives at a criterion for a machine to be decomposable (Theorem 4) and also develops a simple technique for the application of this criterion, leading either to the result that a given machine is not decomposable or alternatively to the specifications of the component machines.

The problem studied in this paper is closely connected with the state assignment problem for sequential machines discussed by Hartmanis in a recent paper.[5] However, both the formulation of the problem as well as the proposed method of solution are rather different.

II. Basic Definitions

Following Mealy,[6] Aufenkamp and Hohn,[7] and Ginsburg,[8] we define a (deterministic, completely specified,

* Received by the PGEC, March 2, 1961. The research work for this paper was carried out at the Elec. Engrg. Dept., Syracuse University, Syracuse, N. Y.

† Technion, Israel Institute of Technology, Haifa, Israel.

[1] E. F. Moore, "Gedanken-experiments on sequential machines," in "Automata Studies," Princeton University Press, Princeton, N. J., pp. 129–153; 1956.

[2] J. Hartmanis, "Symbolic analysis of a decomposition of information processing machines," *Information and Control*, vol. 3, pp. 154–178; June, 1960.

[3] G. Birkhoff, "Lattice Theory," Am. Math. Soc. Coll. Publ., vol. 25, New York, N. Y., rev. ed., pp. 21–24; 1948.

[4] V. S. Krishnan, "The theory of homomorphisms and congruences for partially ordered sets," *Proc. Indian Acad. Sci.*, Sec. A, vol. 22, pp. 1–19; 1945.

[5] J. Hartmanis, "On the state assignment problem for sequential machines. I," IRE Trans. on Electronic Computers, vol. EC-10, pp. 157–165; June, 1961.

[6] G. H. Mealy, "A method of synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1079; September, 1955.

[7] D. D. Aufenkamp and F. E. Hohn, "Analysis of sequential machines," IRE Trans. on Electronic Computers, vol. EC-6, pp. 276–285; December, 1957.

[8] S. Ginsburg, "A synthesis technique for minimal state sequential machines," IRE Trans. on Electronic Computers, vol. EC-8, pp. 13–24; March, 1959.

synchronous, sequential) machine M as a system consisting of

1) a finite set of *states* $S=\{s_1, \cdots, s_n\}$, $n>1$,
2) a finite set of *inputs* $X=\{x_1, \cdots, x_m\}$, $m>1$,
3) a finite set of *outputs* $Z=\{z_1, \cdots, z_r\}$, $r>1$,
4) a function Λ (*output* function) associating with each state s_i and each input x_j and output z_k. We shall use the (operational) notation $z_k=s_i\Lambda x_j$.
5) a function Δ (*next state* function) associating with each (s_i, x_j)-pair a state $s_l=s_i\Delta x_j$.

Methods of describing sequential machines by tables (flow table, Δ-, Λ-matrix)[8] or by weighted graphs (flow diagrams) are well known. Given the initial state of such a machine M, and any input sequence, the method of deriving the corresponding output sequence is also assumed to be known.

In discussing different machines M, M', etc., unprimed, primed, etc., symbols will be understood to refer to M, M', etc., respectively. However, following usual operational notations, symbols Δ and Λ will be used throughout for any machine.

Following Moore,[1] two machines M and M' will be said to be *isomorphic* ($M\cong M'$) if the Δ- and Λ-matrices of M can be obtained from the corresponding M'-matrices by substituting new names for the states, wherever they occur.

In other words, M and M' are isomorphic if $X=X'$, $Z=Z'$, and there exists a one-one correspondence $s\leftrightarrow s'$ between S and S' such that $s\leftrightarrow s'$ and $x=x'$ implies

$$s\Delta x \leftrightarrow s'\Delta x' \tag{1a}$$

as well as

$$s\Lambda x = s'\Lambda x'. \tag{1b}$$

A formal definition of the cascade-connection of machines (see Fig. 1) will now be given. Let M' and M'' be two machines with $Z'=X''$, *i.e.*, the outputs of M' coincide with the inputs of M''. The machine M is said to be the *cascade-connection* of M' and M'' (notation: $M=M'\bigcirc M''$) if[9]

$$X = X', \qquad Z = Z'', \qquad S = S'\times S'',$$

and for any $x=x'\in X$ and any $s=(s', s'')\in S$,

$$s\Delta x = (s'\Delta x', s''\Delta x'') \tag{2a}$$

$$s\Lambda x = s''\Lambda x'', \tag{2b}$$

where

$$x'' = s'\Lambda x'. \tag{2c}$$

A machine M will be called *decomposable* if there exist two machines M', M'', such that

$$M \cong M' \circ M''.$$

[9] $S'\times S''$ denotes the Cartesian product of S' and S'', *i.e.*, the set of all ordered pairs (s', s''), where s' runs through S' and s'' through S''.

III. Transition Graphs

For the purpose of studying machine compositions (and decompositions), it will be convenient to introduce the concept of transition graph. We define a *transition graph* as a finite, directed graph G with exactly one edge leaving each vertex.[10] We shall use the notation $v_iG=v_k$ to indicate that G contains an edge from vertex v_i to vertex v_j ("takes v_i into v_k").

Given a machine M with inputs $x_1, \cdots, x_m$ and states $s_1, \cdots, s_n$ we associate with any input x_j a transition graph G_j consisting of n vertices $v_1, \cdots, v_n$ such that

$$v_iG_j = v_k \quad \text{whenever} \quad s_i\Delta x_j = s_k. \tag{3}$$

The transition graphs of a machine thus obviously correspond to the transition matrices introduced by Seshu, Miller and Metze.[11]

For an example see Table I specifying a two-input, three-state machine M', and Figs. 4(b) and 4(d) showing the corresponding transition graphs. The following is easily verified.

Lemma 1

Each maximal connected subgraph of a transition graph includes exactly one cycle.[12]

To apply transition graphs to machine decomposition, we also have to introduce the following definition.

Let G and $\overline{G}$ be transition graphs and $v\rightarrow\bar{v}$ a many-one correspondence between their vertices,[13] such that

$$v \rightarrow \bar{v} \quad \text{implies } vG \rightarrow \bar{v}\overline{G}$$

or

$$\overline{vG} = \bar{v}\overline{G}. \tag{4}$$

The correspondence $v\rightarrow\bar{v}$ is a *homomorphism* between G and $\overline{G}$. Examples of homomorphisms are indicated in Fig. 4 by dotted lines (G_1 and G_1'; G_2 and G_2'). Now, let $M=M'\bigcirc M''$, and let G_j and G_j' be the transition graphs of M and M' associated with the same input element $x_j=x_j'$ (see Tables I-III, and Fig. 4). Furthermore let the vertex v of G_j represent $s\in S$ and the vertex v' of G_j' represent s', where $s=(s', s'')$. Considering the correspondence $v\rightarrow v'$, we have from (2a)

$$vG_j \rightarrow v'G_j',$$

i.e., the correspondence $v\rightarrow v'$ defines a homomorphism between G_j and G_j' for any $j=1, \cdots, m$.

[10] For graph-theoretical terminology see S. Seshu and M. B. Reed, "Linear Graphs and Electrical Networks," Addison and Wesley Publishing Co., Reading, Mass.; 1961.

[11] S. Seshu, *et al.*, "Transmission matrices of sequential machines," IRE Trans. on Circuit Theory, vol. CT-6, pp. 5–12; March, 1959.

[12] A cycle is defined as a sequence of vertices $(v_1, \cdots, v_k)$ without repetitions, such that $v_iG=v_{i+1}(i=1, \cdots, k-1)$ and $v_kG=v_1$.

[13] *I.e.*, each vertex v of G has one and only one correspondent vertex $\bar{v}$ of $\overline{G}$, whereas, conversely, each vertex $\bar{v}$ of $\overline{G}$ is the correspondent of at least one vertex v of G.

TABLE I

Δ- AND λ-MATRICES OF MACHINE M'

s' \ x'	1	2
1	2	2
2	1	3
3	2	1

$s'\Delta x'$

s' \ x'	1	2
1	1	2
2	2	1
3	1	1

$s'\lambda x'$

TABLE II

Δ-MATRIX OF MACHINE M''

s'' \ x''	1	2
1	1	2
2	2	1

$s'\Delta x''$

TABLE III

Δ-MATRIX OF $M = M' \bigcirc M''$

s \ x	1	2
(1, 1)	(2, 1)	(2, 2)
(1, 2)	(2, 2)	(2, 1)
(2, 1)	(1, 2)	(3, 1)
(2, 2)	(1, 1)	(3, 2)
(3, 1)	(2, 1)	(1, 1)
(3, 2)	(2, 2)	(1, 2)

$s\Delta x$

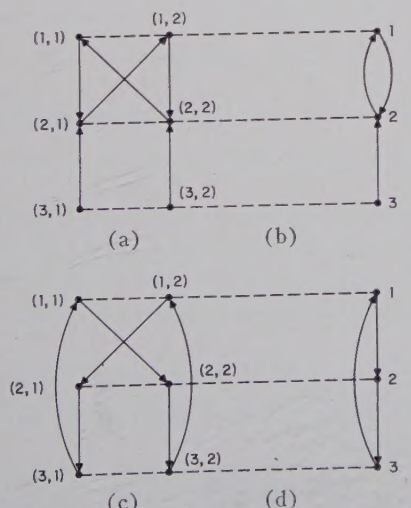


Fig. 4—Transition graphs of machines M (Table III) and M' (Table I). (a) Transition graph G_1 of M. (b) Transition graph G_1' of M'. (c) Transition graph G_2 of M. (d) Transition graph G_2' of M'.

Furthermore, transition graphs of isomorphic machines associated with the same input element are obviously isomorphic. The above results, obtained for $M = M' \bigcirc M''$, will, therefore, also hold if $M \cong M' \bigcirc M''$. Thus we have the following.

Theorem 1

Let $M \cong M' \bigcirc M''$ and let G_j and G_j' be the transition graphs of M and M', associated with the same input element $x_j = x_j'$. Furthermore, let the state s of M correspond to the state (s', s'') of $M' \bigcirc M''$, s and s' being represented by the vertices v and v' of G_j and G_j', respectively. Then $v \rightarrow v'$ defines a homomorphism between G_j and G_j'.

The relationship between machine decomposition and homomorphic transition graphs, established in Theorem 1, justifies a further study of transition-graph homomorphisms. We shall find especially that Theorems 2 and 3, which follow, are directly applicable to machine decomposition.

Let us consider (see Fig. 4) the cycle (1, 1), (2, 1), (1, 2), (2, 2) of G_1. Its homomorphic image is the 1, 2 cycle of G_1'. Similarly, the 6-vertex cycle G_2 is mapped into the 3-vertex cycle G_2', any two vertices of G_2 with distance 3, being mapped into the same vertex of G_2'. Generalizing, one easily verifies the following.

Theorem 2

Let $v \rightarrow \bar{v}$ be a homomorphism between the transition graphs G and $\overline{G}$. Then this correspondence will map any cycle C of G of length ρ into a cycle $\overline{C}$ of $\overline{G}$ of length $\bar{\rho}$, with $\rho = k\bar{\rho}$ (k an integer). Any two vertices of the G-cycle with distance $\bar{\rho}$ are mapped into the same vertex of $\overline{G}$. The following is rather evident.

Lemma 2

The homomorphic image of a connected transition graph is connected.

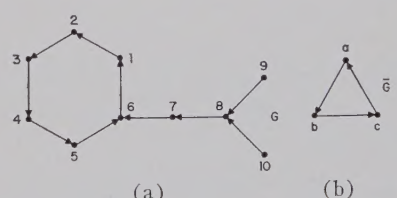


Fig. 5—Example of homomorphic transition graphs.

Consider now the transition graph G shown in Fig. 5(a). Let us assume that $\overline{G}$ is a homomorphic image of G containing a cycle a, b, c into which the 1-6 cycle of G is mapped as follows:

$$1, 4 \rightarrow a; \qquad 2, 5 \rightarrow b; \qquad 3, 6 \rightarrow c.$$

If $\overline{G}$ consists only of the 3 vertices a, b, c [Fig. 5(b)], we must have $7 \rightarrow b$. This simply follows from $6 \rightarrow c$, $7G = 6$, and the fact that only b is taken into c by $\overline{G}$. Similarly, we must have $8 \rightarrow a$, $9, 10 \rightarrow c$. If $\overline{G}$ would contain, in addition to the a, b, c cycle, another vertex d,

we could not have $d\overline{G}=d$, due to Lemma 2. Thus, no vertex is taken by $\overline{G}$ into d, and therefore only vertices 9 and 10 could be mapped into d. Applying the previous reasoning, we must, therefore, still have

$$7 \rightarrow b, \qquad 8 \rightarrow a,$$

whereas 9 and 10 could be mapped either into d or into c.

The above method of reasoning leads to the following.

Theorem 3

Let, under the assumptions of Theorem 2,

$$C = (a_1, \cdots, a_\rho)$$
$$\overline{C} = (\bar{a}_1, \cdots, \bar{a}_{\bar{\rho}}).$$

Let $P=(b_1, \cdots, b_\sigma)$ be an oriented path of G with only vertex $a_\rho=b_\sigma$ common to both C and P (see Fig. 6).

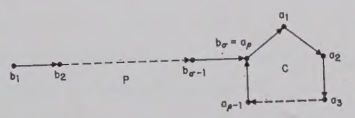


Fig. 6—Illustrating Theorem 3.

Denoting by $\bar{n}$ the total number of vertices of $\overline{G}$, let b_μ be a vertex of P with

$$\mu > \bar{n} - \bar{\rho}.$$

Then

$$\bar{b}_\mu = \bar{a}_\lambda,$$

where

$$\equiv \mu - \sigma \pmod{\bar{\rho}}.$$

A formal proof of Theorem 3 follows. First, we prove the following.

Lemma 3

Let ν, $1 \leqq \nu \leqq \sigma$ be the minimal integer with $\bar{b}_\nu \in \overline{C}$.

Then $\bar{b}_1, \cdots, \bar{b}_{\nu-1}$ are all different. Indeed, let γ be maximal such that

$$\bar{b}_\gamma = \bar{b}_\delta, \qquad 1 \leqq \gamma < \delta \leqq \nu - 1.$$

We would then also have

$$\bar{b}_{\gamma+1} = \bar{b}_{\delta+1}.$$

This, however, would, for $\delta<\nu-1$, contradict the assumption that γ is maximal, and for $\delta=\nu-1$, the assumption that v is minimal. Lemma 3 is thus proved.

Proof of Theorem 3

It follows from Lemma 3 that $\bar{n} \geqq \bar{\rho}+\nu-1$. Hence, from

$$\mu > \bar{n} - \bar{\rho},$$

we have

$$\mu \geqq \nu,$$

whence

$$\bar{b}_\mu \in \overline{C}.$$

Now let

$$\bar{b}_\mu = \bar{a}_\lambda,$$

then (writing $\bar{b}\overline{G}^2$ for $(\bar{b}\overline{G})\overline{G}$, etc., as usual)

$$\bar{b}_\sigma = \bar{b}_\mu \overline{G}^{\sigma-\mu} = \bar{a}_\lambda \overline{G}^{\sigma-\mu}$$

also

$$\bar{b}_\sigma = \bar{a}_\rho = \bar{a}_{\bar{\rho}},$$

hence

$$\lambda + \sigma - \mu \equiv 0 \pmod{\bar{\rho}}$$

or

$$\lambda \equiv \mu - \sigma \pmod{\bar{\rho}}.$$

Theorem 3 is thus proved.

IV. Partitions

"Admissible partitions" as defined later in this Section will also have an important role in connection with machine decompositions.[14]

A *partition* π of a set S is a collection $\{A, B, C, \cdots\}$ of subsets of S, such that each element of S belongs to one and only one of the subsets of the collection. We shall use the notation $s_i \equiv s_j(\pi)$ to indicate that elements s_i, s_j of S belong to the same subset of S, under the partition π.

If all the subsets of a finite set S, forming the partition π, have the same number of elements, the partition will be called *uniform*.

Let G and $\overline{G}$ be homomorphic transition graphs under the correspondence $v \rightarrow \bar{v}$. We define a partition π of V, the set of vertices of G, by

$$v_i \equiv v_j(\pi) \quad \text{whenever} \quad \bar{v}_i = \bar{v}_j. \tag{5}$$

This partition π has the following property:

$$v_i \equiv v_j(\pi) \quad \text{implies} \quad v_i G \equiv v_j G(\pi). \tag{6}$$

Indeed, $v_i \equiv v_j(\pi)$ implies, by (5), $\bar{v}_i = \bar{v}_j$, hence $\bar{v}_i\overline{G} = \bar{v}_j\overline{G}$. Applying (4) and (5), we obtain $v_iG \equiv v_jG(\pi)$.

Given a transition graph G, a partition π of its vertices will be said to be *admissible* (by G) if it has the property (6).

Given a machine M, a partition π of its set of states S will be called *admissible* if, for any input $x \in X$, and any two states s, $t \in S$

$$s \equiv t(\pi) \quad \text{implies} \quad s\Delta x \equiv t\Delta x(\pi).$$

[14] The term "admissible partition" used here corresponds to "partition having the substitution property" in Hartmanis.[2,5]

For any machine M, a given partition π of S obviously defines corresponding partitions of the vertices of the transition graphs of M. Evidently, the partition π of S is admissible if, and only if, it is admissible by all the transition graphs of M. Now, let $M \cong M' \circ M''$, with

$$s \leftrightarrow (s', s''), \quad t \leftrightarrow (t', t'')[s, t \in S; s', t' \in S'; s'', t'' \in S''].$$

We define a partition π of S by

$$s \equiv t(\pi) \quad \text{whenever} \quad s' = t'.$$

If follows immediately from (2a) that π is admissible. It is also evident that π is uniform.

Conversely, given a machine M and a partition π of S, admissible and uniform, we now wish to show that M is decomposable.

For this purpose, let us arrange the elements of S in matrix form:

$$\begin{matrix} s_{11}s_{12} \cdots s_{1n''} \\ \cdots\cdots\cdots \\ s_{n'1}s_{n'2} \cdots s_{n'n''}, \end{matrix}$$

the n' rows of this matrix constituting the n' π-subsets of S. We assume π to be nontrivial, *i.e.*, $1 < n' < n$. We now define a machine M' as follows:

$$X' = X$$
$$S' = \{s_1', \cdots, s_{n'}'\}$$
$$s_\alpha' \Delta x_j' = s_\gamma'$$

whenever

$$s_{\alpha\beta} \Delta x_j = s_{\gamma\delta}. \tag{7}$$

The partition π being admissible, the definition of γ by (7) will not depend on β, *i.e.*, s_γ' is uniquely defined by s_α' and x_j'.

The output set Z' of M' we define to consist of the $n'm$ elements $z_{\alpha j}'$, $\alpha = 1, \cdots, n'$; $j = 1, \cdots, m$, with the output function of M' being given by

$$s_\alpha' \Lambda x_j' = z_{\alpha j}'.$$

Next, we define M'' by

$$X'' = Z', \qquad S'' = \{s_1'', \cdots, s_{n''}''\}, \qquad Z'' = Z.$$

With $x_{\alpha j}'' = z_{\alpha j}'$, $\alpha = 1, \cdots, n'$, $j = 1, \cdots, m$ denoting an arbitrary input element of M'', we define

$$s_\beta'' \Delta x_{\alpha j}'' = s_\delta'',$$

where δ is determined by (7) and

$$s_\beta'' \Lambda x_{\alpha j}'' = s_{\alpha\beta} \Lambda x_j.$$

One immediately verifies that

$$M \cong M' \circ M''.$$

Thus we arrive at the following.

Theorem 4

A machine M is decomposable if, and only if, there exists a nontrivial, uniform, admissible partition π of S, the set of states of M.

V. Machine Decompositions

We have seen in Section IV that for a given machine M, the establishment of a (nontrivial) uniform and admissible partition π of S is essential for the decomposition of M into the cascade connection of two smaller machines, M' and M''. Once such a partition π has been found, the procedure described in Section IV may be followed to define M' and M''. In this Section we wish to show how such a partition π may be found, using the theory of homomorphic transition graphs developed in Section III. We shall best illustrate our method by following an example. Consider the Δ-matrix of a machine M, given in Table IV. The corresponding transition graphs are shown in Fig. 7. We have $n = 8$, and therefore, we must have either $n' = 2$ or $n' = 4$. Let us consider $n' = 2$ first, *i.e.*, we would have to partition $S = \{1, \cdots, 8\}$ into two subsets containing four elements each, the partition to be admissible by $G_1, \cdots, G_4$ (Fig. 7).

Applying Theorem 2 to the cycle (3, 7, 8, 4) of G_1, we must have $\bar{\rho} = 1$ or 2 and for both cases

$$\bar{4} = \bar{7}, \tag{8}$$

TABLE IV

The Δ-Matrix of a Machine M, to be Decomposed

s \ x	1	2	3	4
1	2	6	2	6
2	5	1	5	1
3	7	2	7	8
4	3	7	6	7
5	6	1	3	1
6	7	2	1	2
7	8	3	2	6
8	4	7	4	7

$s\Delta x$

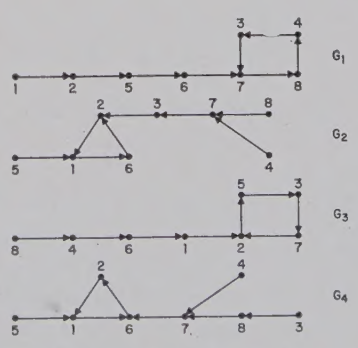


Fig. 7—The transition graphs of M (Table IV).

$\bar{v}$ denoting the image of v. But applying (4) and (8) to G_2 would lead to

$$\bar{4} = \bar{7} = \bar{3} = \bar{2} = \bar{1} = \bar{6},$$

which would contradict the requirement of having four elements in each subset.

We next consider $n' = 4$. The above argument indicates that

$$\bar{3}, \bar{7}, \bar{8}, \bar{4} \tag{9}$$

are all different. If we now apply Theorem 3 to G_1 (with $\bar{n} = n' = 4$) we find

$$\bar{6} = \bar{3}, \qquad \bar{5} = \bar{4}, \qquad \bar{2} = \bar{8}, \qquad \bar{1} = \bar{7}. \tag{10}$$

We thus conclude that $\pi\{\{1, 7\}, \{2, 8\}, \{3, 6\}, \{4, 5\}\}$ is the only possibility for the partition in question. One now easily verifies that π does indeed satisfy all the requirements of Theorem 4, and thus will lead to a decomposition of M.

On the other hand, let us assume that one of the transition graphs of an eight-state machine M, to be decomposed, contains a cycle C of 5 vertices. Applying Theorem 2, we must either have $\bar{\rho} = 5$ or $\bar{\rho} = 1$. In the first case we would have $n' = 5$, in the second case $n'' = 5$. Evidently no such nontrivial uniform partition exists, *i.e.*, M is not decomposable.

VI. Conclusions

This study leads to a graph-theoretical method for the decomposition of a sequential machine into cascaded components whenever possible. However, a number of additional problems arise in this connection, the solutions of which would no doubt lead to considerable engineering advantages as to the synthesis of large sequential machines.

The extension of this study to incompletely specified machines appears rather important. Also, if a machine M is specified, the engineer will frequently build a machine M^*, which can do more than the machine M originally specified ($M^* \geqq M$, in the notation of Ginsburg[8]). The decomposition concept should be generalized accordingly.

Furthermore, more complicated decompositions of a machine into a number of smaller components should be investigated, following to some extent the various decomposition schemes discussed by Ashenhurst[15] for combinational networks.

Evidently, suitable coding of the inputs and outputs may lead to simpler realizations of specified machines, *e.g.*, to a reduction in the number of interconnecting leads of partially cascaded machines (Fig. 3). Further study should, therefore, also cover the input- and output-assignment problem in addition to the state-assignment problem.

One would also have to investigate the possibility of programming (for available digital computers) a decomposition algorithm based on the method described.

Acknowledgment

The author is much indebted to Dr. Sundaram Seshu, for valuable discussions and suggestions in connection with the preparation of this paper.

[15] R. L. Ashenhurst, "The decomposition of switching functions," *Proc. Internatl. Symp. on the Theory of Switching*, April 2–5, 1957, in *Ann. Computation Lab.*, Harvard University, Cambridge, Mass., vol. 29, pp. 74–116; 1959.

On the State Assignment Problem for Sequential Machines II*

R. E. STEARNS† AND J. HARTMANIS†

Summary—The object of this paper is to find state assignments for the internal states of a sequential machine such that the logical equations representing the machine are relatively simple. This is done by finding assignments for which the computation of a particular state variable depends only on the previous values of a small subset of the variables.

The chief tool is the concept of a partition pair, which describes (loosely speaking) the information going into and resulting from the evaluation of a state variable. A necessary and sufficient condition for the existence of assignments with reduced dependence is found in terms of these pairs, and their algebraic properties are worked out so that they can be handled and generated. It is shown that the same methods can also be used to find input and output assignments with reduced dependence.

The case of "don't care" conditions is considered and the theory is seen to apply, except that the failure of an algebraic property makes it weaker.

Introduction

THE purpose of this paper, as in Part I,[1] is to choose (binary) variables describing the internal states of a finite-state sequential machine so that the logical relationships among them are relatively simple. This is achieved by assigning state variables so that each variable (at time $t+1$) depends only on a small subset of the variables (at time t). Although we recognize the fact that such assignments are not always best, we feel that they are generally far better than most assignments and often best. Hence, we believe that such an investigation is worthwhile.

Throughout the paper, it will be assumed that the reader is familiar with Part I, or else that he knows the basic properties of partitions.

The chief tool of this paper will be the concept of a partition pair which generalizes the notion of a partition with substitution property used by Hartmanis.[1,2] It will be seen that many of the results of Part I[1] are special cases of the more general theorems derived in this report.

Using the notation of Hartmanis,[1] we shall let $\boldsymbol{M}$ be a finite-state sequential machine with internal states $S_1, \cdots, S_n$, inputs $I_1, \cdots, I_m$, and outputs $O_1, \cdots, O_n$ (the outputs are not necessarily all distinct). The behavior of $\boldsymbol{M}$ is given by a flow table (Fig. 1). The left column gives the present state, the top row the inputs, and the table entries give the next state. The last row labeled Z indicates the output for each state.

When variables $y_1, y_2, \cdots, y_k$ have been assigned to designate the internal states of the machine, then the relations between the values of the variables at the present state and those of the next state are expressed by:

$$y_i = f_i(y_1, \cdots, y_k, I),$$

an abbreviation of

$$y_i(t+1) = f_i[y_1(t), \cdots, y_k(t), I(t)].$$

To show the need for a concept more general than that of a partition with S.P. (substitution property)[1] and to illustrate the general direction of this paper, we consider the sequential machine $\boldsymbol{A}$ given in Fig. 2.

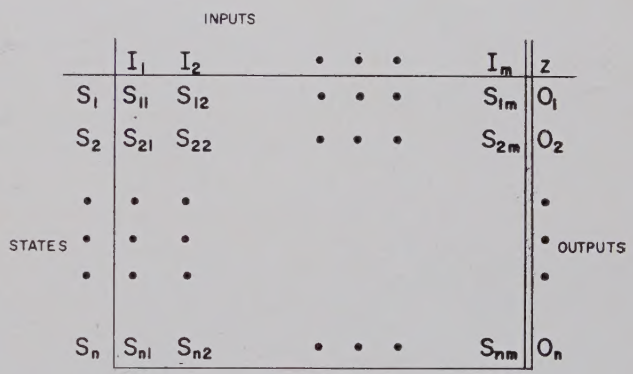


Fig. 1—Flow table for a sequential machine.

INPUTS

STATES	I_1	I_2	I_3	I_4	z
1	1	2	3	4	1
2	3	4	1	2	1
3	2	1	4	3	0
4	4	3	2	1	0

OUTPUTS

Fig. 2—Sequential machine $\boldsymbol{A}$.

Machine $\boldsymbol{A}$ is seen to have one nontrivial partition with S.P., namely $\{\overline{1, 4}; \overline{2, 3}\}$. We use this partition in making the assignment shown in Fig. 3. (In this example we have assumed that we can also assign the input codes. A discussion of input assignments will appear at the end of the paper.) The relations between the old and

* Received by the PGEC, February 28, 1961; revised manuscript received, August 28, 1961.

† General Electric Res. Lab., Schenectady, N. Y.

[1] J. Hartmanis, "On the state assignment problem for sequential machines. I," IRE Trans. on Electronic Computers, vol. EC-10, pp. 157–165; June, 1961.

[2] J. Hartmanis, "Symbolic analysis of a decomposition of information processing machines," *Inform. and Control*, vol. 3, pp. 154–178; June, 1960.

new state variables are given by:

$$y_1 = \bar{x}_1 y_1 + x_1 \bar{y}_1$$

$$y_2 = \bar{x}_2 y_1 \bar{y}_2 + \bar{x}_2 \bar{y}_1 y_2 + x_2 \bar{y}_1 \bar{y}_2 + x_2 y_1 y_2$$

$$z = \bar{y}_2.$$

As predicted by Hartmanis,[1] y_1 can be calculated from itself and the inputs. The inputs have also been assigned to reduce dependence. Even so, the expression for y_2 is rather complex because there are not any further partitions with S.P. to reduce the dependence of y_2. Next we will look at an assignment given in Fig. 4 which does not use the partition with S.P.

The equations for this assignment are:

$$y_1 = \bar{x}_1 y_2 + x_1 \bar{y}_2$$

$$y_2 = \bar{x}_2 y_1 + x_2 \bar{y}_1$$

$$z = \bar{y}_1.$$

Both expressions are simple; y_1 depends on y_2 and x_1 and y_2 depends on y_1 and x_2. This can also be seen from Fig. 5 which shows a realization of $\boldsymbol{A}$ using this assignment.

	y_1	y_2
1 →	0	0
2 →	1	0
3 →	1	1
4 →	0	1

STATES

	x_1	x_2
I_1 →	0	0
I_2 →	1	0
I_3 →	1	1
I_4 →	0	1

INPUTS

Fig. 3—First assignment for machine $\boldsymbol{A}$.

	y_1	y_2
1 →	0	0
2 →	0	1
3 →	1	0
4 →	1	1

STATES

	x_1	x_2
I_1 →	0	0
I_2 →	0	1
I_3 →	1	0
I_4 →	1	1

INPUTS

Fig. 4—Second assignment for machine $\boldsymbol{A}$.

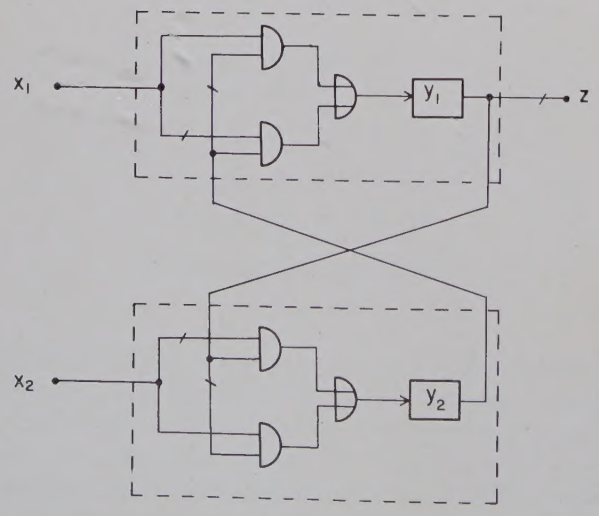


Fig. 5—Realization of machine $\boldsymbol{A}$ using second assignment.

Notice how the machine is almost decomposed into two independent units,[2] except that the wires from memory have been crossed. One might call such a machine "cross decomposed."

To understand the basic ideas utilized in making this assignment, observe that the first variable y_1 is constant for the states 1 and 2 and for states 3 and 4. Thus this variable induces the partition $\pi_1 = \pi(y_1) = \{\overline{1, 2}; \overline{3, 4}\}$ on the set of states of the machine $\boldsymbol{A}$. Similarly, the second variable defines the partition $\pi_2 = \pi(y_2) = \{\overline{1, 3}; \overline{2, 4}\}$. Neither of these partitions has the substitution property. On the other hand, it can be seen from the equations for the second assignment (or the flow table for this machine), that if we know in which block of π_1 the state of the machine is contained (that is the value of y_1) then, for any input we can compute (y_2) the block of π_2 in which the next state of the machine will be contained. The same statement holds if we interchange π_1 and π_2. In other words, if there is some ignorance about the present state of the machine and we know only in which block of π_1 the state is contained, then the ignorance about the next state of the machine is given by π_2, because we can compute only the block of π_2 which contains the next state of the machine. We shall refer to such partitions, describing the information flow in a sequential machine, as partition pairs on the machine, and they will be the basic tool in this investigation. In the following sections, we shall give precise meaning to these ideas, develop the mathematical properties of partition pairs, and show their use in the study of sequential machines.

Partition Pairs

Definition 1:

A partition π on a set S is a collection of disjoint subsets of S such that their union is S. These disjoint subsets are called the blocks of π and are thought of as equivalence classes on the set S.

Definition 2:

A partition pair (π, π') on the states of a sequential machine $\boldsymbol{M}$ is an ordered pair of partitions on the set of states such that if S_i and S_j belong to the same block of π, then for each input I, IS_i and IS_j are in the same block of π'. (IS_i is the state the machine goes into from S_i when input I is applied.)

Thus, π gives the rows which we demand be identified in the flow table and π' sets up equivalence classes sufficient to accomplish these demands. This implies that, if (π, π') is a partition pair and we know only the block of π in which the state of the machine is contained, then we can determine for any input the block of π' in which the next state of the machine will be contained. Observe that if $\pi = \pi'$, then this definition reduces to the definition of a partition with the S.P.[1] It follows at once from Definition 2 that:

Lemma 1: If (π, π') is a partition pair for a sequential

machine M and $\pi_0 \leq \pi$ and $\pi_0' \geq \pi'$, then (π_0, π') and (π, π_0') are also partition pairs.

This merely says that we can decrease the demands or increase the equivalences and we will still have equivalences sufficient for our demands.

To illustrate these ideas, consider machine B given in Fig. 6. For this machine

$$(\tau_1, \tau_1') = (\{\overline{1,3}; \overline{2}; \overline{4}; \overline{5}; \overline{6}\}, \{\overline{1}; \overline{2}; \overline{3,5}; \overline{4,6}\})$$

is a partition pair because the identification of the states 1 and 3 requires that the states 3, 5, and 4, 6 are identified and this identification is sufficient. It can be seen from the flow table or Lemma 1 that

$$(\tau_2, \tau_2') = (\{\overline{1,3}; \overline{2}; \overline{4}; \overline{5}; \overline{6}\}, \{\overline{1,2}; \overline{3,5}; \overline{4,6}\})$$

is also a partition pair. Finally, note that the states 2 and 4 can be put in the same block of τ_1 without imposing any addition equivalences. Thus, we see that

$$(\tau_3, \tau_3') = (\{\overline{1,3}; \overline{2,4}; \overline{5}; \overline{6}\}, \{\overline{1}; \overline{2}; \overline{3,5}; \overline{4,6}\})$$

is also a partition pair for this machine.

STATES	INPUTS x_1x_2 = 00	01	11	z (OUTPUTS)
1	6	1	3	0
2	5	2	4	0
3	4	1	5	0
4	3	2	6	0
5	2	3	5	1
6	1	4	6	0

Fig. 6—Machine B.

The reader may return to the discussion of machine A and verify that (π_1, π_2) and (π_2, π_1) are partition pairs. The next lemma shows that partition pairs on a machine can be combined component-wise by the partition operations to generate new partition pairs on the machine. The proof of this and the following lemma are given in the Appendix.

Lemma 2: If (π, π') and (τ, τ') are partition pairs for a sequential machine M, then so are $(\pi \cdot \tau, \pi' \cdot \tau')$ and $(\pi + \tau, \pi' + \tau')$. (The algebra of partitions is discussed in Hartmanis[1,2] and Birkhoff.[3]

Before we can give the theorems relating independence of variables and partition pairs, we need one additional concept.

Definition 3:

If π is a partition on the states of a machine M, let $M(\pi) = \Sigma \pi_i$ where the sum is over the set of π_i such that (π_i, π) is a partition pair. Similarly, define $m(\pi) = \Pi \pi_i$ where the product is over the π_i such that (π, π_i) is a partition pair.

[3] G. Birkhoff, "Lattice theory," *Am. Math. Soc. Colloquium Publ.*, vol. 25; 1948.

Lemma 3: If π is a partition on the set of states of a machine, then $M(\pi)$ and $m(\pi)$ exist and $[M(\pi), \pi]$ and $[\pi, m(\pi)]$ are partition pairs.

Thus, $M(\pi)$ is the largest partition τ such that (τ, π) is a partition pair and $m(\pi)$ is the smallest partition such that (π, τ) is a pair.

An easy application of Lemma 1 to Definition 3 shows that the M and m operations are Monotone: $\pi \leq \tau$ implies that $M(\pi) \leq M(\tau)$ and $m(\pi) \leq m(\tau)$.

Returning to the discussion about machine B of Fig. 6, we observe that $M(\tau_3') = \tau_3$ and $m(\tau_3) = \tau_3'$. We shall refer to such pairs later as Mm pairs, and their importance and properties will be discussed.

Partition Pairs and Assignments with Reduced Dependence

The object of this section is to show that the notion of dependence of variables is equivalent to the holding of certain algebraic relations (information flow inequalities) among partitions associated with the variables. When we say that a variable depends only on a certain set of variables, we mean that the variable can be calculated from this set and the input independently of the value of the other state variables. We do not claim, however, that this set is unique.

Theorem 1: Suppose the states of a machine M are assigned in variables $y_1, \cdots, y_k$. For each i, $1 \leq i \leq k$, let $\pi(y_i)$ be the state partition whose elements are in the same block if and only if they are assigned the same value of y_i. If $y_i(t+1)$ depends only on the input and $\{y_j(t) \mid j \in P_i\}$ (P_i is some subset of indexes), then

$$\prod_{j \in P_i} \pi(y_j) \leq M[\pi(y_i)].$$

This relation will be called an "information flow inequality." Its meaning is quite simple if one recalls that the largest partition (least amount of information about the present state) from which we can compute the block of $\pi(y_i)$ in which the next state of the machine is contained is given by $M[\pi(y_i)]$. The inequality simply states that if we can compute y_i for the next state, then we must have at least as much information about the present state as is contained in $M[\pi(y_i)]$. A formal proof of this theorem is given in the Appendix.

Example: In the case of machine A, assignment two, we have

$$\pi(y_2) = M[\pi(y_1)] = \{\overline{1,3}; \overline{2,4}\}, \; P_1 = \{2\},$$

and

$$\pi(y_1) = M[\pi(y_2)] = \{\overline{1,2}; \overline{3,4}\}, \; P_2 = \{1\}.$$

Theorem 2: If there is a set $\{(\pi_i, \pi_i')\}$ of partition pairs for a machine M, and for each i, there is some set P_i such that $\Pi_{j \in P_i} \pi_j' \leq \pi_i$; then there is a state assignment such that each π_i' is associated with a subset Y_i of the variables and the variables in Y_i at time $t+1$ depend only on the values of the variables in $\cup_{j \in P_i} Y_j$ at time t.

Proof: We assign the variables in Y_i so that they are constant over the states in each block of π_i', but are different for states in different blocks. If the variables are binary, then Y_i will have $\lceil \log_2 \#(\pi_i) \rceil$ variables, where $\lceil x \rceil$ is the smallest integer not less than x and $\#(\pi)$ is the number of blocks in π.[1]

To show that this yields an assignment with the required independence between the variables, note that if we know the values of the variables in $\bigcup_{j \in P_i} Y_j$, then we know the block of $\prod_{j \in P_i} \pi_j'$ which contains the state. But this tells us the block of π_i by the inequality $\prod_{j \in P_i} \pi_j' \leq \pi_i$ and hence, the block of π_i' into which the machine goes by the definition of a partition pair. Since the variables in Y_i are constant on this block, their values have been determined from the values of the variables in $\bigcup_{j \in P_i} Y_j$ as was to be shown.

Note: If $\{(\pi_i, \pi_i')\}$ satisfy the conditions of Theorem 2, then so does $\{[M(\pi_1'), \pi_i']\}$ since $\pi_i \leq M(\pi_1')$. Considering the sets Y_i as single (not necessarily binary) variables, Theorem 2 gives us a converse to Theorem 1. Thus, *$\prod_{j \in P_i} \pi_j' \leq M(\pi_i')$ is a necessary and sufficient condition for the existence of a state assignment with the Y_i depending only on the Y_j, j in P_i.*

The total number of binary digits required to take advantage of the partition structure described in Theorem 2 is less than or equal to

$$\lceil \log_2 n(\prod \pi_i') \rceil + \sum_i \lceil \log_2 \#(\pi_i') \rceil,$$

where $n(\pi)$ is the number of elements in the largest block of π. (That this expression may yield an over estimate is shown by machine C in Hartmanis.[1]) The first term of this expression gives the extra variables required to distinguish between states whose Y_i assignments are the same if $\prod \pi_i' \neq 0$.

Returning to the machine ***B*** of Fig. 6, we see that

$$(\pi_0, \pi_0') = (\{\overline{1,2}; \overline{3,4}; \overline{5,6}\}, \{\overline{1,2}; \overline{3,4}; \overline{5,6}\})$$
$$(\pi_1, \pi_1') = (\{\overline{1,3,5}; \overline{2,4,6}\}, \{\overline{1,3,5}; \overline{2,4,6}\})$$
$$(\pi_2, \pi_2') = (\{\overline{1,2,3,4}; \overline{5,6}\}, \{\overline{1,2}; \overline{3,4,5,6}\})$$
$$(\pi_3, \pi_3') = (\{\overline{1,2}; \overline{3,4,5,6}\}, \{\overline{1,2,3,4}; \overline{5,6}\})$$

are partition pairs for this machine. To obtain an assignment with simple logical relations between the variables, we shall apply Theorem 2 using the last three partition pairs. It is seen that $\pi_1' \leq \pi_1$, $\pi_3' \leq \pi_2$, $\pi_2' \leq \pi_3$, and $\pi_1' \cdot \pi_2' \cdot \pi_3' = 0$, the partition with one element blocks. Since π_1', π_2', and π_3' have two blocks each, we know that there is a three-digit assignment according to π_1', π_2', and π_3' so that $y_1 = f_1(y_1, I)$, $y_2 = f_2(y_3, I)$ and $y_3 = f_3(y_2, I)$. One such assignment is given in Fig. 7. The equations relating the variables of the old and new states are:

$$y_1 = \bar{x}_2 \bar{y}_1 + x_2 y_1$$
$$y_2 = \bar{x}_2 y_3 + x_2 \bar{y}_3 + x_1$$
$$y_3 = \bar{x}_2 \bar{y}_2 + x_1 y_2.$$

Note: Theorem 2 assures us of these functional independences (y_1 independent of y_2 and y_3, etc.) when making transitions for the six states in the flow table. It is only with careful filling of the don't care conditions that these Boolean equations, which define an eight state machine, also exhibit this independence. We shall see later (Theorem 5) when don't care conditions are discussed that this can always be done.

By assigning variables to distinguish between the blocks of π_2' and π_3', we have automatically assigned a pair of variables which distinguish between the blocks of $\pi_2' \cdot \pi_3'$. Since $\pi_2' \cdot \pi_3' = \pi_0' = \pi_0$ and $\pi_1 = \pi_1'$, we see that these partitions have S.P. Thus, this assignment (Fig. 7) uses the two partitions π_1 and π_0 which have S.P. to reduce the dependence between the variables,[1] but it further assigns the variables to the blocks of π_0 in a way that is dependence reducing and cannot be done with partitions with S.P. The realization of this machine is shown in Fig. 8.

STATES	y_1	y_2	y_3
1 →	0	0	0
2 →	1	0	0
3 →	0	1	0
4 →	1	1	0
5 →	0	1	1
6 →	1	1	1

Fig. 7—State assignment for machine ***B***.

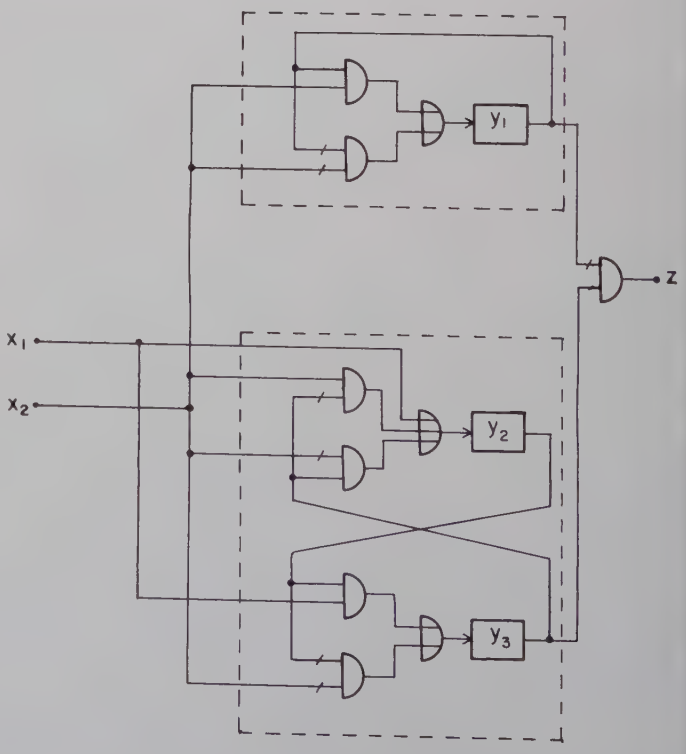


Fig. 8—A realization of machine ***B***.

The *Mm* Pairs

In this section we shall define a special class of partition pairs, the *Mm* pairs, and derive their basic algebraic properties which will enable us to generate and manipulate the pairs for the analyses of sequential machines. The importance of the *Mm* pairs will be evident after we show in this section that all other partition pairs on a sequential machine can be generated from the *Mm* pairs. Since the set of all *Mm* pairs on a sequential machine is generally a lot smaller than the set of all partition pairs, the *Mm* pairs yield a more compact way of representing the structural information about a sequential machine.

Definition 4:

A partition pair (π, π') for a machine $\boldsymbol{M}$ will be called an *Mm* pair if and only if $\pi = M(\pi')$ and $\pi' = m(\pi)$.

This definition states that in a *Mm* pair (π, π'), π is the largest partition (least amount of information) from which we can compute π' and, at the same time, π' is the finest partition (most information) that can be determined from π. We shall now define a partial ordering for partition pairs which is helpful in visualizing structural properties of machines.

Definition 5:

If (π, π') and (τ, τ') are partition pairs for a machine $\boldsymbol{M}$, then write $(\pi, \pi') \leq (\tau, \tau')$ if and only if $\pi \leq \tau$ and $\pi' \leq \tau'$.

If we restrict ourselves to *Mm* pairs on a machine, then the ordering defined above can be determined from the first or second components of the *Mm* pairs as stated below.

Lemma 4: If (π, π') and (τ, τ') are *Mm* pairs for a machine $\boldsymbol{M}$, then $\pi \leq \tau$ if and only if $\pi' \leq \tau'$.

To verify this result we recall that, if $\pi \leq \tau$, then the monotonicity of the m operation implies that $m(\pi) \leq m(\tau)$. Since for *Mm* pairs, $m(\pi) = \pi'$ and $m(\tau) = \tau'$, this inequality implies that $\pi' \leq \tau'$. A similar argument shows that $\pi' \leq \tau'$ implies $\pi \leq \tau$.

The next lemma gives a further property of the M and m operations and is important because of the two corollaries which follow.

Lemma 5: If π is a partition on machine $\boldsymbol{M}$, then

$$M\{m[M(\pi)]\} = M(\pi) \text{ and } m\{M[m(\pi)]\} = m(\pi).$$

Proof: $[M(\pi), \pi]$ and $\{M(\pi), m[M(\pi)]\}$ are partition pairs by Lemma 3, and therefore (definition of m) $m[M(\pi)] \leq \pi$. By monotonicity $M\{m[M(\pi)]\} \leq M(\pi)$. On the other hand, $\{M(\pi), m[M(\pi)]\}$ is a partition pair, and hence, $M\{m[M(\pi)]\} \geq M(\pi)$. Thus the first equality holds. The second follows by a similar argument.

From Lemma 5 and Definition 3 we obtain the next result.

Corollary 1: For all partitions π on a machine $\boldsymbol{M}$, $\{M(\pi), m[M(\pi)]\}$ and $\{M[m(\pi)], m(\pi)\}$ are *Mm* pairs.

If for a machine $\boldsymbol{M}$, $\pi = M(\tau)$ for some τ, we shall call π a M-partition. Similarly, if $\pi = m(\tau)$, we call π a m-partition. Corollary 1 says that each M-partition (and also each m-partition) belongs to some *Mm* pair.

We see from the previous results that if (π, π') is a partition pair on a machine $\boldsymbol{M}$, then

$$\{M(\pi'), m[M(\pi')]\} \quad \text{and} \quad \{M[m(\pi)], m(\pi)\}$$

are *Mm* pair on this machine. From this observation the next result follows:

Corollary 2: Given a partition pair (π, π') for machine $\boldsymbol{M}$, there exists an *Mm* pair (τ, τ') such that $\tau \geq \pi$ and $\tau' \leq \pi'$.

Corollary 2 says that any partition pair can be obtained from some *Mm* pair (τ, τ') by enlarging τ' and refining τ. For an illustration, we return to the discussion of machine $\boldsymbol{B}$ following Definition 2 and observe that (τ_1, τ_1') is obtained from *Mm* pair (τ_3, τ_3') by refining τ_3 to τ_1 and enlarging (trivially) τ_3' to τ_1'. Recall that Lemma 1 showed the converse, that any such enlarging of τ' and refining of τ results in a partition pair. Thus we see that the set of all *Mm* pairs describes the set of partition pairs and, in general, the set of *Mm* pairs is much smaller than the set of partition pairs.

We now continue with the study of the algebraic properties of *Mm* pairs. We shall derive methods for combining the *Mm* pairs and show that the set of all *Mm* pairs on a machine forms a lattice under the ordering of Definition 5. It is important to observe that a component-wise combination of two *Mm* pairs by the partition operations will, according to Lemma 2, yield a partition pair, but that this does not have to be an *Mm* pair. We can, however, as stated in the next two lemmas, add m-partitions to get m-partitions and multiply M-partitions to get M-partitions. This will be useful for generating *Mm* pairs. The proofs of these lemmas and Theorem 3 appear in the Appendix.

Lemma 6:

If $\pi = M(\pi')$ and $\tau = M(\tau')$, then $\pi \cdot \tau = M(\pi' \cdot \tau')$.

Lemma 7:

If $\pi' = m(\pi)$ and $\tau' = m(\tau)$, then $\pi' + \tau' = m(\pi + \tau)$.

These two lemmas make it easy to prove the interesting algebraic properties stated in the following theorem.

Theorem 3: The set of all *Mm* pairs for a machine $\boldsymbol{M}$ under the order of Definition 5 forms a lattice. If (π, π') and (τ, τ') are *Mm* pairs, then,

$$\text{g.l.b.}\{(\pi, \pi'), (\tau, \tau')\} = [\pi \cdot \tau, m(\pi \cdot \tau)]$$

and

$$\text{l.u.b.}\{(\pi, \pi'), (\tau, \tau')\} = [M(\pi' + \tau'), \pi' + \tau'].$$

We shall now discuss the computation of the *Mm* pairs to be used in the study of sequential machines.

Definition 6:

If a and b are states of a machine $\boldsymbol{M}$, then let π_{ab} be the partition on the states of $\boldsymbol{M}$ which identifies a and b; but leaves the other states in one element blocks.

Theorem 4: If (π, π') is a Mm pair, then $\pi' = \Sigma m(\pi_{ab})$, where the sum is taken over all π_{ab} such that $\pi_{ab} \leq \pi$.

Proof: Since $\pi \geq \pi_{ab}$, (π_{ab}, π') is a pair (Lemma 1) and $m(\pi_{ab}) \leq \pi'$. Hence, $\Sigma m(\pi_{ab}) \leq \pi'$. But

$$[\textstyle\sum \pi_{ab},\ \sum m(\pi_{ab})] = [\pi,\ \sum m(\pi_{ab})]$$

is a partition pair by Lemma 2. Therefore, $\Sigma m(\pi_{ab}) \geq m(\pi) = \pi'$ and the theorem is proved.

Theorems 3 and 4 are important for purposes of computation. First we find the set $\{m(\pi_{ab})\}$, a process which takes $(1/2)n(n-1)$ calculations. Then we take all possible sums of these partitions. Theorem 3 assures us that these sums are m-partitions and Theorem 4 assures us that we can find all m-partitions this way. The M-partition π for m-partition π' can be found by inspecting the flow table or by using the obvious relation: $\pi = \Sigma \pi_{ab}$ sum over $\{\pi_{ab} | m(\pi_{ab}) \leq \pi'\}$. In other words, sum over the π_{ab} such that (π_{ab}, π') is a partition pair; we are just merging the rows of the flow table that are the same under the equivalence relation π'.

The Analysis of a Machine

We shall now apply the principles developed above to an analysis of machine $\boldsymbol{C}$ given in Fig. 9.

First we calculate the $m(\pi_{ab})$. To find $m(\pi_{12})$, we look at the flow table and see that to identify rows 1 and 2, we must set states 1 and 3, 1 and 5 equivalent and by transitivity, then we must also have 3 and 5 equivalent. Thus

$$m(\pi_{12}) = \{\overline{1, 3, 5}; \overline{2}; \overline{4}\}.$$

Continuing, we arrive at the following list of $m(\pi_{ab})$ for machine $\boldsymbol{C}$.

$$\begin{aligned}
m(\pi_{12}) &= \{\overline{1, 3, 5}; \overline{2}; \overline{4}\} = \pi_1' \\
m(\pi_{13}) &= m(\pi_{45}) = \{\overline{1, 3, 4}; \overline{2, 5}\} = \pi_2' \\
m(\pi_{14}) &= m(\pi_{35}) = \{\overline{1}; \overline{2}; \overline{3, 5}; \overline{4}\} = \pi_3' \\
m(\pi_{15}) &= m(\pi_{23}) = m(\pi_{25}) = m(\pi_{34}) = I \\
m(\pi_{24}) &= \{\overline{1, 5}; \overline{2}; \overline{3}; \overline{4}\} = \pi_4'.
\end{aligned}$$

Now that we have the $m(\pi_{ab})$, we can forget about the flow table and work only with these partitions.

To complete the list of m-partitions for machine $\boldsymbol{C}$ we consider all possible sums of the $m(\pi_{ab})$. Adding non-empty subsets of them, we do not get any additional partitions. Summing over the empty set, we get (by definition) the zero partition. This completes our list of m-partitions. Now we start work on the M-partitions:

$$M(\pi_1') = \sum \pi_{ab} = \pi_{12} + \pi_{14} + \pi_{39} + \pi_{24},$$

summed over $m(\pi_{ab}) \leq \pi_1'$.

Continuing, we get all the M-partitions:

$$\begin{aligned}
M(I) &= I \\
M(\pi_1') &= \pi_1 = \{\overline{1, 2, 4}; \overline{3, 5}\} \\
M(\pi_2') &= \pi_2 = \{\overline{1, 3}; \overline{2}; \overline{4, 5}\} \\
M(\pi_3') &= \pi_3 = \{\overline{1, 4}; \overline{3, 5}; \overline{2}\} \\
M(\pi_4') &= \pi_4 = \{\overline{1}; \overline{2, 4}; \overline{3}; \overline{5}\} \\
M(0) &= 0.
\end{aligned}$$

Combining the previous results we obtain a complete list of the Mm pairs and these contain the information about the information flow in the machine. The Mm lattice is given in Fig. 10 and the Mm pairs are the following:

$$\begin{aligned}
&(I, I), \qquad (0, 0) \\
&(\pi_1, \pi_1') = (\{\overline{1, 2, 4}; \overline{3, 5}\}, \{\overline{1, 3, 5}; \overline{2}; \overline{4}\}) \\
&(\pi_2, \pi_2') = (\{\overline{1, 3}; \overline{2}; \overline{4, 5}\}, \{\overline{1, 3, 4}; \overline{2, 5}\}) \\
&(\pi_3, \pi_3') = (\{\overline{1, 4}; \overline{3, 5}; \overline{2}\}, \{\overline{1}; \overline{2}; \overline{3, 5}; \overline{4}\}) \\
&(\pi_4, \pi_4') = (\{\overline{1}; \overline{2, 4}; \overline{3}; \overline{5}\}, \{\overline{1, 5}; \overline{2}; \overline{3}; \overline{4}\}).
\end{aligned}$$

It is interesting to observe that the partitions with S.P. are easily obtained from this list. If π has S.P., then $M(\pi) \geq \pi \geq m(\pi)$. In this case, π can satisfy $\pi_3 \geq \pi \geq \pi_3'$, $I \geq \pi \geq I$, or $0 \geq \pi \geq 0$. Thus there are two nontrivial partitions with S.P., and they are $\{\overline{1, 4}; \overline{2}; \overline{3, 5}\}$ and $\{\overline{1}; \overline{2}; \overline{3, 5}; \overline{4}\}$.

STATES	X_1X_2 = 00	01	11	10	z
1	3	1	4	2	0
2	1	5	4	2	0
3	3	4	3	5	0
4	5	1	4	2	0
5	5	4	3	5	1

OUTPUTS

Fig. 9—Machine $\boldsymbol{C}$.

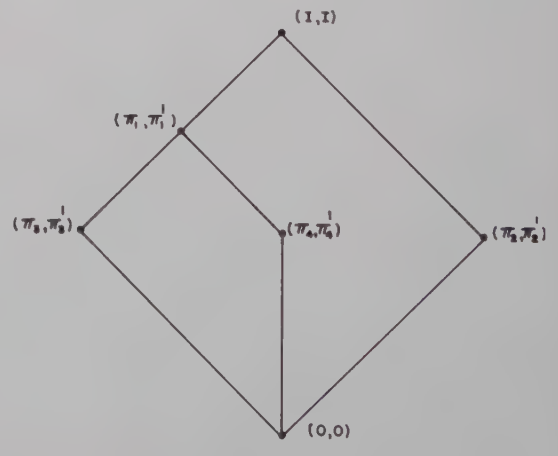


Fig. 10—Mm lattice for machine $\boldsymbol{C}$.

Now we shall try to apply our knowledge about the information flow in machine **C** to obtain a good state assignment. Our objective is to make a three-digit binary state assignment reducing the number of dependences. In terms of partitions, we are looking for three partitions τ_1, τ_2, τ_3 of two blocks each such that

$$\tau_1 \cdot \tau_2 \cdot \tau_3 = 0$$

and $M(\tau_1)$, $M(\tau_2)$, $M(\tau_3)$ are each greater than the product of only a small subset of the partitions τ_1, τ_2, τ_3. [Recall that $M(\tau_i)$ is the largest partition, or the least amount of information, from which we can compute the block of τ_i in which the next state of the machine is contained.] Generally speaking, the larger the M-partitions, the better chance of this happening. We observe, for instance, that if a variable according to some partition τ is to depend only on a variable according to some other partition τ', then $\tau \leq M(\tau')$ (Theorem 1) and $M(\tau')$ can have at most two blocks. In the case of machine **C**, the only such M-partitions are π_1 and I.

With this in mind, we decide to let $\tau_3 = \pi_1$ and later we will choose τ_1 to be a two-block partition that is an enlargement of π_1'; hence, (τ_3, τ_1) will be a partition pair and the state variable according to τ_1 will depend only on the state variable according to τ_3 and the input. The largest m-partition less than τ_3 is π_3', and hence $M(\tau_3) = \pi_3'$.

We see that if we choose $\tau_2 = \pi_2$, then $\tau_2 \cdot \tau_3 \leq M(\tau_3)$ and we have additional reduced dependence. Finally, we have to choose $\tau_1 > \pi_2'$ so that $\tau_1 \cdot \tau_2 \cdot \tau_3 = 0$ and this can be done if states "1" and "4" are separated. Letting $\tau_1 = \{\overline{1, 3, 5}; \overline{2, 4}\}$, we get the inequality $\tau_1 \cdot \tau_2 \leq M(\tau_2)$ and τ_2 is independent of τ_3. This leads to an assignment requiring 42 diodes.

An alternative choice for τ_2 would be $\{\overline{1, 5}; \overline{2, 3, 4}\}$ which gives the relation $\tau_2 \cdot \tau_3 \leq M(\tau_2)$ and then enlarging π_1' to $\tau_1 = \{\overline{1, 3, 4, 5}; \overline{2}\}$ which gives $\tau_1 \cdot \tau_2 \cdot \tau_3 = 0$ and $\tau_1 \cdot \tau_3 \geq M(\tau_3)$. This leads to an assignment found by T. A. Dolotta, shown in Fig. 11 along with the corresponding partition pairs and information-flow inequalities. It is interesting to observe that for either choice, the number of dependencies is the same. In the first case, the realization has the form shown in Fig. 12(a) and in the second case, the form shown in Fig. 12(b). Both choices utilize Mm pairs which are high on the Mm lattice. The actual logical equations for Dolotta's assignment are given below:

$$y_1 = x_1 \bar{x}_2 \bar{y}_3$$

$$y_2 = x_1 \bar{y}_3 + x_2 y_3 + \bar{x}_1 y_2 y_3 + \bar{x}_2 \bar{y}_2 \bar{y}_3$$

$$y_3 = x_1 y_3 + \bar{x}_1 \bar{x}_2 \bar{y}_1 + \bar{x}_1 x_2 y_1$$

$$z = y_2 y_3.$$

It is seen that this assignment requires only 30 diodes. For the sake of comparison Dolotta found a bad assignment that requires 65 diodes.

$(M(\tau_1), \tau_1) = (\{\overline{1, 2, 4}; \overline{3, 5}\}, \{\overline{1, 3, 4, 5}; \overline{2}\})$

$(M(\tau_2), \tau_2) = (\{\overline{1}; \overline{2, 4}; \overline{3; 5}\}, \{\overline{1, 5}; \overline{2, 3, 4}\})$

$(M(\tau_3), \tau_3) = (\{\overline{1, 4}; \overline{3, 5}; \overline{2}\}, \{\overline{1, 2, 4}; \overline{3, 5}\})$

$M(\tau_1) \geq \tau_3$	
$M(\tau_2) \geq \tau_2 \cdot \tau_3$	
$M(\tau_3) \geq \tau_1 \cdot \tau_3$	
$\tau_1 \cdot \tau_2 \cdot \tau_3 = 0$	

	y_1	y_2	y_3
$1 \rightarrow 0$	0	0	
$2 \rightarrow 1$	1	0	
$3 \rightarrow 0$	1	1	
$4 \rightarrow 0$	1	0	
$5 \rightarrow 0$	0	1	

Fig. 11—Partition pairs, information-flow inequalities and state assignment for machine **C**.

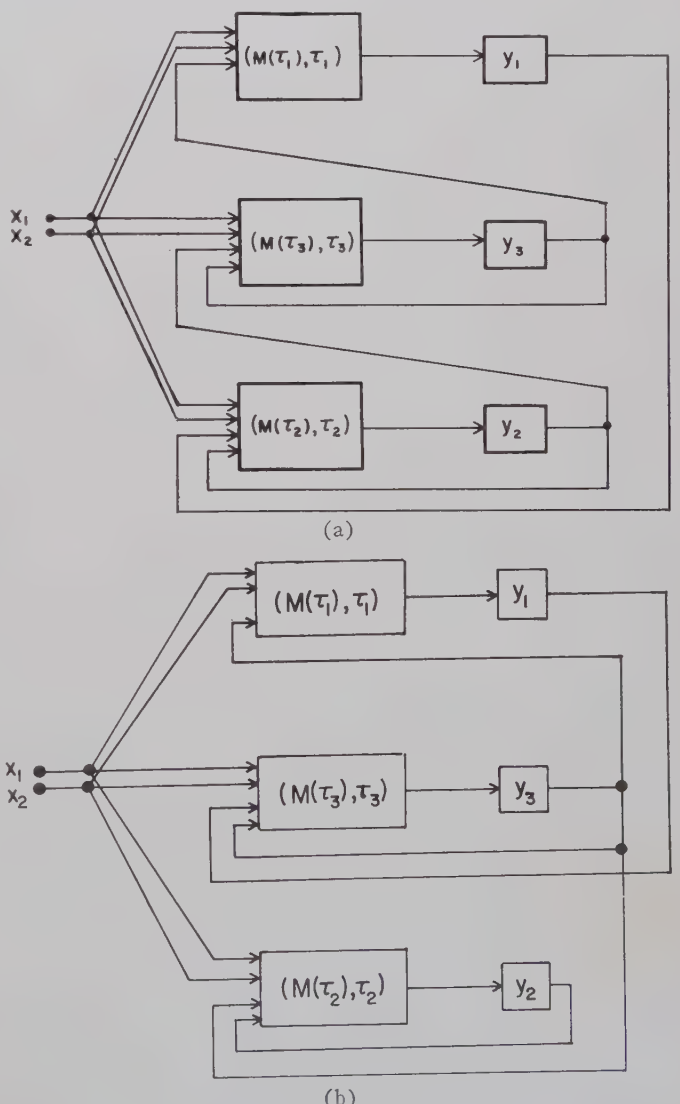


Fig. 12—(a) Realization for machine **C** resulting from first choice of τ_2. (b) Realization for machine **C** resulting from assignment of Fig. 11.

Don't Care Conditions

A sequential machine with "Don't Care" (D.C.) conditions is given by a flow table with some of the entries left blank. (See Figs. 13 and 14 for examples.) The machine designer is left with the option of assigning values to the blank entries.

The notion of a partition pair extends to such machines if we change the phrase "IS_i and IS_j are in the same block" of Definition 2 into "IS_i and IS_j, if they have been specified, are in the same block."

In Lemma 2, the proof that $(\pi \cdot \tau, \pi' \cdot \tau')$ is a partition pair goes through by making the same change of wording. The proof that $(\pi + \tau, \pi' + \tau')$ is a partition pair, however, cannot be made to go through as the counter example in Fig. 13 shows.

For machine ***D***, (π_1, π_1) and (π_2, π_2) are partition pairs, where $\pi_1 = \{\overline{1, 2}; \overline{3}; \overline{4}\}$ and $\pi_2 = \{\overline{1}; \overline{2, 3}; \overline{4}\}$, but $(\pi_1 + \pi_2, \pi_1 + \pi_2)$ is not a partition pair. Note also that π_1 and π_2 are furthermore partitions with S.P.,[1] but $\pi_1 + \pi_2$ does not have the substitution property.

X

STATES	0	1	z	
1	4	1	1	
2	-	2	0	OUTPUTS
3	1	3	1	
4	2	3	0	

Fig. 13—Machine ***D***.

X_1X_2

STATES	00	01	11	10	z	
1	1	6	-	3	1	
2	5	2	4	-	1	
3	2	5	-	4	1	
4	6	1	3	-	0	OUTPUTS
5	3	-	6	1	0	
6	-	4	2	5	0	

Fig. 14—Machine ***E***.

$M(\pi)$ and $m(\pi)$ are still well defined but $[M(\pi), \pi]$ need not be a partition pair as that part of Lemma 3 was based on the part of Lemma 2 which now fails. In other words, the set $\{\pi \mid (\pi, \pi') \text{ is a partition pair}\}$ has no l.u.b. For machine ***D***, $(\pi, \{\overline{1}; \overline{2, 3}; \overline{4}\})$ is a partition pair for $\pi = \{\overline{1}; \overline{2, 3}; \overline{4}\}$ and $\pi = \{\overline{1}; \overline{2, 4}; \overline{3}\}$, but not for $\pi = \{\overline{1}; \overline{2, 3, 4}\}$ which is their sum.

Theorem 1 remains true, but since $[M(\pi), \pi]$ may not be a partition pair, the theorem does not give a converse for the new form of Theorem 2 (Theorem 5).

We shall now study the problem of filling in the D.C. conditions in order to be able to apply the previously developed theory.

For a given set of partition pairs on a machine, it may not be always possible to fill in the D.C. conditions in the flow table with the states of the machine in such a way that the partition pairs still hold for the new machine. For example, the D.C. condition in machine ***D*** cannot be filled with any of the four states of the machine without destroying one of the two partition pairs,

$$(\pi_1, \pi_1) \quad \text{or} \quad (\pi_2, \pi_2).$$

The following lemma gives a condition on the partition pairs under which all existing D.C. conditions can be filled with the states of the machine.

Definition 7:

The set of partitions $\{\pi_i\}$, $i = 1, 2, \cdots, k$, on a set S is said to be permutable if and only if the intersection $\bigcap_1^k B_i$ is nonvoid where B_i is any block of π_i.

Lemma 8: If $\{(\pi_i, \pi_i')\}$, $i = 1, \cdots, k$, is a set of partition pairs on the set of states S of a sequential machine M with D.C. conditions and the $\{\pi_i'\}$ are permutable, then the D.C. conditions can be filled with states from S in such a way that $\{(\pi_i, \pi_i')\}$ are partition pairs on the resulting machine.

Proof: We shall show that one of the D.C. conditions can be filled preserving the pairs (π_i, π_i') and the lemma will follow by induction. Suppose condition IS_j needs to be filled in. Let B_i be the block of π_i containing S_j. If IS_k is specified for any state S_k in B_i, let B_i' be the block of π_i' containing IS_k. (Of course all such S_k go into the same block.) If IS_k is not defined for any S_k in B_i, choose B_i' arbitrarily. Now let IS_j be any element in $\bigcap B_i'$. Such an element exists because $\{\pi_i'\}$ are permutable and we see that the (π_i, π_i') are still partition pairs by the choice of B_i'. Q.E.D.

The previous lemma is quite strong because it states that under the given conditions, all possible sets of D.C. conditions can be filled with the states of the machine. We can loosen the conditions of the lemma and get a necessary and sufficient condition that a given D.C. condition can be filled preserving a set of partition pairs.

Corollary 3: Let $\{(\pi_i, \pi_1')\}$, $i = 1, 2, \cdots, k$, be a set of partition pairs for a machine with D.C. conditions. Then the D.C. condition for IS_j can be filled with a state of the machine, so that the given partition pairs are preserved if, and only if, the nonvoid blocks B_1' of π_i', into which the blocks B_i (of π_i) containing S_j are mapped, have a nonvoid intersection, $\bigcap B_i' \neq \phi$.

The proof follows along the same lines as the proof of the previous lemma.

Ordinarily, when a set of partition pairs has been chosen that satisfies certain information-flow inequalities which suggest a reduced dependence assignment, the permutability condition of Lemma 8 is not satisfied. We shall show, however, that the machine and partitions can be augmented in such a way that the partition pairs and information flow inequalities are preserved and the permutability condition satisfied. To ease the reader into the next lemma and theorem, we shall first illustrate the procedure on a specific example, machine ***E*** of Fig. 14.

Let $\pi_1 = \{\overline{1, 2, 3, 4}; \overline{5, 6}\}$, $\pi_2 = \{\overline{1, 2, 5, 6}; \overline{3, 4}\}$, and $\pi_3 = \{\overline{1, 3, 5}; \overline{2, 4, 6}\}$. Then (π_3, π_1), (π_1, π_2), and (π_2, π_3) are partition pairs for $\boldsymbol{E}$ and if there were no D.C. conditions to fill, we could expliot these pairs for a three variable state assignment so that y_1 depended only on y_3; y_2 depended only on y_1, and y_3 on y_2. There is, however, no way of filing the flow table blanks with the states of the machine so that this partition-pair structure is preserved. To see this, apply the results of the previous corollary.

We shall now use the three partitions π_1, π_2, and π_3 to make an assignment with reduced dependence which will also introduce additional states in such a way that the partition pairs are preserved. We shall give the theorems justifying this method after this example.

The assignment exploiting π_1, π_2, and π_3 is given in Fig. 15.

There are values of these state variables (when $y_1 = y_2 = 1$) for which no state has been assigned. We see that this binary assignment has, in a natural way, named two more states which do not appear in the flow table. These extra states can be added to the flow table leaving all new entries blank. Fig. 16 results.

This new machine $\boldsymbol{E}'$ has partition pairs (τ_3, τ_1), (τ_1, τ_2), and (τ_2, τ_3) where $\tau_1 = \{\overline{1, 2, 3, 4}; \overline{5, 6, 7, 8}\}$, $\tau_2 = \{\overline{1, 2, 5, 6}; \overline{3, 4, 7, 8}\}$, and $\tau_3 = \{\overline{1, 3, 5, 7}; \overline{2, 4, 6, 8}\}$. The relations among the τ_i are the same as among the π_i [*i.e.*, $\pi_3 \leq M(\pi_1)$ and $\tau_3 \leq M(\tau_1)$, etc.]. The assignment by the π_i carries over to the τ_i since the blocks of τ_i are in reality blocks of π_i augmented with extra states. Now that the state variables are allowed their full range of values, the D.C. conditions can be filled (Fig. 16 becomes Fig. 17) so that (τ_3, τ_1), (τ_1, τ_2) and (τ_2, τ_3) remain partition pairs. We get the following equations with the predicted dependencies:

$$y_1 = \bar{x}_2 y_3 + x_2 \bar{y}_3$$
$$y_2 = \bar{x}_1 y_1 + x_1 \bar{y}_1$$
$$y_3 = \bar{x}_2 y_2 + x_2 \bar{y}_2.$$

We shall show that such a procedure can always be applied.

Lemma 9: Suppose $\boldsymbol{M}$ is a sequential machine with D.C. conditions such that

1) $\{\pi_i, \pi_i')\}$, $i = 1, \cdots, k$, is a set of partition pairs on $\boldsymbol{M}$;
2) for each i, there is an associated set of numbers P_i such that $\prod_{j \in P_i} \pi_j' \leq \pi_i$;
3) $\Pi_1^k \pi_i' = 0$.

Suppose also that the state variables in the set Y_i are assigned according to π_i', $i = 1, \cdots, k$ (the sets Y_i, Y_j are disjoint for $i \neq j$); then $\boldsymbol{M}$ extends naturally to a machine M' with D.C. conditions whose states are the set of all m-tuples of variables in Y_i, $i = 1, 2, \cdots, k$. The sets of variables in Y_i induce partitions τ_i on $\boldsymbol{M}'$ such that $\{\tau_i\}$ are permutable, $\Pi_1 \tau_i = 0$, and $\{(\prod_{j \in P_i} \tau_j, \tau_i)\}$ are partition pairs on $\boldsymbol{M}$.

	y_1	y_2	y_3
1 →	0	0	0
2 →	0	0	1
3 →	0	1	0
4 →	0	1	1
5 →	1	0	0
6 →	1	0	1

STATES

Fig. 15—Assignment for machine $\boldsymbol{E}$.

	x_1x_2 = 00	01	11	10	z
1 (000)	1	6	-	3	1
2 (001)	5	2	4	-	1
3 (010)	2	5	-	4	1
4 (011)	6	1	3	-	0
5 (100)	3	-	6	1	0
6 (101)	-	4	2	5	0
7 (110)	-	-	-	-	-
8 (111)	-	-	-	-	-

STATES — OUTPUTS

Fig. 16—Machine $\boldsymbol{E}'$.

	x_1x_2 = 00	01	11	10	z
(000) 1	1	6	8	3	1
(001) 2	5	2	4	7	1
(010) 3	2	5	7	4	1
(011) 4	6	1	3	8	0
(100) 5	3	8	6	1	0
(101) 6	7	4	2	5	0
(110) 7	4	7	5	2	-
(111) 8	8	3	1	6	-

STATES — OUTPUTS

Fig. 17—Machine $\boldsymbol{E}'$ with D.C. state conditions filled.

Proof: The machine $\boldsymbol{M}'$ is obtained as $\boldsymbol{E}'$ was obtained from $\boldsymbol{E}$ by adding a new state for each m-tuple of variables in Y_i, $i = 1, 2, \cdots, k$, that was not assigned to a state of $\boldsymbol{M}$ and then filling the new entries of the flow table with D.C. conditions.

We shall now show that the partitions τ_i, induced by identifying all states which do not differ on the variables in Y_i, are permutable. To see this, observe that picking a block B_i of τ_i corresponds to fixing the values of the variables in Y_i. Thus, picking a block from each τ_i corresponds to fixing all the variables in $Y_1, Y_2, \cdots, Y_k$; the state corresponding to this m-tuple is the one and only state contained in the intersection of B_i, $i = 1, 2, \cdots, k$. This shows that the τ_i are permutable and (because of condition 3) that

$$\prod_1^k \tau_i = 0.$$

Finally, suppose $I(Y_1^0, \cdots, Y_k^0)$ and $I(Y_1^1, \cdots, Y_k^1)$ are both specified by the flow table, where Y_i^0 and Y_i^1 denote fixed sets of values for the binary variables in Y_i. Then $(Y_1^0, \cdots, Y_k^0)$ and $(Y_1^1, \cdots, Y_k^1)$ must be states of $\boldsymbol{M}$, say S_k and S_l, since all the flow table entries are unspecified for the extra states. If S_k and S_l are in the same block of $\Pi_{P_i}\tau_j$ they are in the same block of $\Pi_{P_i}\pi_j'$, and hence, in the same block of π_i by 2). Since (π_i, π_i') is a partition pair, IS_k and IS_j must be in the same block of π_i', hence in the same block of τ_i. Therefore, $\{(\Pi_{j\in P_i}\tau_j, \tau_i)\}$ are partition pairs. Q.E.D.

The two previous lemmas can be combined to yield the next theorem describing the utilization of partition pairs for machines with D.C. conditions.

Theorem 5: Suppose $\{(\pi_i, \pi_i')\}$, $i=1, \cdots, k$, is a set of partition pairs for a sequential machine $\boldsymbol{M}$ with D.C. conditions such that $\Pi_1^k\pi_i' = 0$ and there is some set of numbers P_i for each i such that $\prod_{j\in P_i}\pi_j' \leq \pi_i$; and suppose a state assignment has been made associating variables in Y_i with π_i' (Y_i disjoint from Y_j if $i \neq j$). Then $\boldsymbol{M}$ can be embedded in a larger sequential machine $\boldsymbol{M}'$ (without D.C. conditions) which has the same inputs as $\boldsymbol{M}$, whose states are given by the variables in Y_i, $i=1, \cdots, k$, and such that the value of the variables in Y_i at time $t+1$ depends only on the value of the variables in $\bigcup_{P_i} Y_j$ at time t.

Proof: We extend $\boldsymbol{M}$ by Lemma 9 and apply Lemma 8 to fill the D.C. entries without destroying the partition pairs $(\Pi_{P_i}\tau_j, \tau_i)$. The D.C. outputs are filled arbitrarily. Theorem 2 assures us that the variables assigned according to the π_i' which are also according to the τ_i, have the desired dependencies. Q.E.D.

The importance of this procedure is that once the partition for the assignment of each set of binary variable has been selected, we can add states in such a manner that the D.C. conditions can be filled preserving the information-flow inequalities and such that no additional variables are required to represent the states.

Thus, when we have a machine with D.C. conditions and we don't even care if they are filled with states not in the flow table (but implicity defined by the assignment variables), we can still use partition pairs to reduce dependencies. The search for useful partition pairs, however, is hampered by the failure of the basic algebraic properties.

Input and Output Assignments

In this section, we wish to explain how the theory of partition pairs developed in the previous sections generalizes to give dependence relations between input and output, state and output, and even input and output. We have already seen in Figs. 2–4 that an expedient choice of input variables along with the choice of state variables can reduce dependence. Even if the inputs and outputs are specified, the machine designer may still use these generalizations to find state partitions which depend on a few of the given inputs or which determine reduced output equations. The basis of these generalizations is that partition pairs are designed to detect functional independence in mappings. Thus, for example, the first partition can be on the set of present inputs and the second partition on the set of next states (here the present states serve as the mappings) or the first partition can be on the set of present states and the second on the set of outputs.

First, we sketch the theory for inputs and states.

Definition 8:

An *I-S* pair (λ, π) on a sequential machine $\boldsymbol{M}$ is an ordered pair of partitions, λ on the set of inputs and π on the set of states, such that if I_i and I_j belong to the same block of λ, then I_iS and I_jS belong to the same block of π for all states S.

This definition is almost the same as that of a partition pair, except that the first partition is on the columns (inputs) of the flow table instead of the rows (states at time t). In both cases, the second partition sets up equivalence among the table entries (states at time $t+1$). Thus these two concepts are the same except that the roles of the rows and columns are interchanged. The following are *I-S* pairs on machine $\boldsymbol{A}$ (Fig. 2): $(\{\overline{I_1, I_2}; \overline{I_3, I_4}\}, \{\overline{1, 2}; \overline{3, 4}\})$, $(\{\overline{I_1, I_3}; \overline{I_2, I_4}\}, \{\overline{1, 3}; \overline{2, 4}\})$, and $(\{\overline{I_1, I_4}; \overline{I_2, I_3}\}, \{\overline{1, 4}; \overline{2, 3}\})$.

Theorem 6: If the words "partition pairs" are changed to "*I-S* pair" in Definition 4, Lemmas 1-7, and Theorems 3 and 4, then these lemmas and theorems still hold.

The proof of this theorem has already been indicated by the previous remarks. The same change of wording makes all the proofs go through except for Lemma 2. In the proof of Lemma 2, we must also substitute I for S, S for I, and make other small changes required for this substitution to make sense. For example, we must change the words "for any input I, IS_i and IS_j" to read "for any state S, I_iS and I_jS."

Note: The lemmas, theorems, and definitions on the algebraic properties of partition pairs after Lemma 2 do not depend directly on the flow table of the machine. Thus, any system of pairs satisfying Lemmas 1 and 2 satisfy the other lemmas and theorems.

The analogs of Theorems 1 and 2 will necessarily be a little different because, in the first case, the names of the rows and the table entries had to be the same (as both were states) whereas in the present case, the columns have separate names. Nevertheless, the analogies are easy to formulate and we shall illustrate with a theorem corresponding to the note after Theorem 2. We use $M^*(\pi)$ to represent the largest λ such that (λ, π) is an *I-S* pair.

Theorem 7: If $\{\pi_i\}$, $i=1, \cdots, n$, are partitions on the states and $\{\lambda_j\}$, $j=1, \cdots, m$, are partitions on the inputs of a sequential machine $\boldsymbol{M}$ such that:

1) State variables in Y_i are assigned according to π_i.
2) Input variables in X_i are assigned according to λ_i.
3) For each $i=1, \cdots, n$, P_i is some subset of the numbers 1 to m.

Then the state variables in Y_i depend only on the input variables in $\bigcup_{j\in P_i} X_j$ (and the other state variables) if and only if

$$\prod_{j\in P_i} \lambda_j \leq M^*(\pi_i).$$

This is proved using arguments similar to those for Theorems 1 and 2.

An illustration of this theorem is provided by Fig. 8 and the equations which precede it. The reader is invited to find the relevant *I-S* pairs and verify this.

Similar results hold for outputs. For example, we can define *S-O* pairs and restate Theorem 7 using state for input and output for state. Returning to machine **C** (Fig. 9), we see that $M'(\{\bar{0};\bar{1}\}) = \{\overline{1,2,3,4};\bar{5}\}$ where M' is analogous to M and M^*. Looking at the assignment and partitions defined in Fig. 11, we see that $\tau_2\cdot\tau_3 \leq M'(\{\bar{0};\bar{1}\})$ and z depends only on y_2 and y_3 as seen in the resulting equation $z=y_2y_3$. The reader may also see from Figs. 2–4 how *S-O* pair $(\{\overline{1,2};\overline{3,4}\}, \{\bar{0};\bar{1}\})$ was used to eliminate a dependence.

If one has a Mealy machine, he may also define an *I-O* pair and look for reduced dependence, but for the Moore machines, as used in this paper, the output is already independent of the input.

Conclusion

In this paper, we have found, in terms of partition pairs, a necessary and sufficient condition for a state assignment (and input assignment) to give reduced dependencies. We have shown that the pairs have some nice algebraic properties, enough to generate them from the π_{ab}. The most difficult step is in choosing a set of partition pairs for the assignment such that the sets P_i (Theorem 2) are small. Nevertheless, this is much easier than trial and error methods and little tricks such as looking for large $M(\pi)$ can be helpful.

When there are "don't care" conditions, the main theorem, when suitably interpreted, still holds but the search for useful partition pairs becomes more difficult because of the breakdown of a basic algebraic property.

Appendix

Proof of Lemma 2

If S_i and S_j are in the same block of $\pi\cdot\tau$, then they are in the same block of π and τ. Therefore, for any input I, IS_i and IS_j are in the same block of π' and τ', and hence, of $\pi'\cdot\tau'$.

If S_i and S_j are in the same block of $\pi+\tau$, there exists a sequence of states $S_{k_0}, S_{k_1}, \cdots, S_{k_n}$, $k_0=i$, $k_n=j$, such that S_{k_r} and $S_{k_{r+1}}$ are in the same block of π for r even, and in the same block of τ for r odd. Therefore, for any input I, IS_{k_r} and $IS_{k_{r+1}}$ are in the same block of π' for r even and the same block of τ' for r odd. Hence, S_i and S_j are in the same block of $\pi'+\tau'$.

Proof of Lemma 3

Let I be the single-block partition and O the partition with one-element blocks. Then (π, I) and (O, π) are seen to be partition pairs. Using the notation of Definition 3 and applying Lemma 2, $[M(\pi), \pi] = (\Sigma_i\pi_i, \Sigma_i\pi)$ and $[\pi, m(\pi)] = (\Pi_i\pi, \Pi_i\pi_i)$ are partition pairs, since the sums and products are taken over nonempty finite sets.

Proof of Theorem 1

The partition $\Pi_{j\in P_i}\pi(y_j)$ identifies all the states which differ only in the variables, not in P_i. Thus, if the states S_k and S_l are in the same block of this partition, then for all inputs I, the states IS_k and IS_l will be in the same block of $\pi(y_i)$, since y_i depends only on the variables y_j, j in P_i, and we conclude that

$$\left[\prod_{j\in P_i}\pi(y_j), \pi(y_i)\right]$$

is a partition pair. From Definition 3, it follows that

$$\prod_{j\in P_i}\pi(y_j) \leq M[\pi(y_i)].$$

Proof of Lemma 6

Suppose $\pi^* = M(\pi'\cdot\tau')$. Since $(\pi\cdot\tau, \pi'\cdot\tau')$ is a partition pair (Lemma 2), we know that $\pi^*\geq\pi\cdot\tau$. By Lemma 1, since $(\pi^*, \pi'\cdot\tau')$ is a pair, so are (π^*, π') and (π^*, τ'). Thus, $\pi^*\leq M(\pi')=\pi$ and $\pi^*\leq M(\tau')=\tau$. Therefore, $\pi^*\leq\pi$ and $\pi^*\leq\tau$, which shows that $\pi^*\leq\pi\cdot\tau$ and thus $\pi^* = \pi\cdot\tau$.

Proof of Lemma 7

Suppose $\pi^* = m(\pi+\tau)$. Since $(\pi+\tau, \pi'+\tau')$ is a partition pair, $\pi^*\leq\pi'+\tau'$. Since $(\pi+\tau, \pi^*)$ is a pair, so are (π, π^*) and (τ, π^*). Hence $\pi^*\geq\pi'$ and $\pi^*\geq\tau'$, which implies that $\pi^*\geq\pi'+\tau'$. Thus $\pi^* = \pi'+\tau'$.

Proof of Theorem 3

$\{M(\pi'\cdot\tau'), m[M(\pi'\cdot\tau')]\}$ is a *Mm* pair by Lemma 5, and $M(\pi'\cdot\tau') = \pi\cdot\tau$ by Lemma 6. Thus by Lemma 4, this pair is a lower bound for (π, π') and (τ, τ'). Since $\pi\cdot\tau$ is the g.l.b. for π and τ, this pair is the g.l.b. for (π, π') and (τ, τ').

A similar proof holds for the second equality using Lemma 7 instead of Lemma 6.

Acknowledgment

The authors owe a debt of gratitude to C. L. Coates for many valuable discussions which have aided in the writing of this report.

Bibliography

[1] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, March, 1954; pp. 275–303, April, 1954.

[2] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, pp. 1045–1079; September, 1955.

[3] E. F. Moore, "Gedanken-Experiments on Sequential Machines," Automata Studies, Princeton, N. J.; 1956.

A Truth Table Method for the Synthesis of Combinational Logic*

SHELDON B. AKERS, JR.†

Summary—This paper describes a method for synthesizing a switching function directly from its truth table. A switching function is defined as *any* mapping of a set of binary input combinations onto 0 and 1. Hence, the procedures apply equally well to the *don't care* cases. The method rests on the concept of *logically passive function* (LPF). Roughly speaking, an LPF is a truth table which can be realized with only AND and OR gates—no inverters. Techniques are described for 1) making a function logically passive, 2) eliminating rows and columns from the truth table of an LPF, 3) synthesizing an LPF in a two-level irredundant form, 4) expanding LPF's and 5) synthesizing LPF's with 3-input majority gates. The underlying methods are quite straightforward and appear to be particularly well suited for mechanization on a digital computer.

I. Introduction

IN this paper, a method will be described for synthesizing combinational logic directly from the truth table of the switching function under consideration. No algebraic or topological manipulations are involved. The term *switching function* will be used in a general sense, *i.e.*, *any* mapping of a set of binary input combinations onto 0 and 1. The truth table for a switching function will have one row for each n-bit input combination in the range of this mapping. While the truth table for the circuit *realization* of a switching function must have 2^n rows (since the circuit defines an output for each of the 2^n input combinations), the table for the function itself may have *any* number of rows (up to 2^n). Thus, if a given function has certain *don't care* input combinations this is denoted implicitly by the fact that these input combinations do not appear as rows in the truth table.

Hereafter, when we speak of a switching function we shall mean the *truth table* which describes that function. Thus two switching functions will be said to be equivalent if and only if their truth tables are identical (within permutation of rows and columns). This is an important consideration.

Consider the switching function of Table I:

TABLE I

A	B	C	F
0	0	0	0
1	0	0	0
1	1	0	1
0	0	1	1
0	1	1	0
1	1	1	1

Such a function is frequently specified by a Veitch diagram such as in Fig. 1 where the x's denote *don't cares*. Now assume a fourth input, $\overline{B}$, were available. Then the corresponding table (Table II) would be:

TABLE II

A	B	C	$\overline{B}$	F
0	0	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	1
0	1	1	0	0
1	1	1	0	1

These two *tables* are certainly *not* equivalent even though both are normally described by the Veitch diagram of Fig. 1. Now consider the function of Table III:

TABLE III

A	B	C	D	F
0	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	1
1	0	1	0	0
1	1	1	0	1

Its Veitch diagram would be that shown in Fig. 2. Certainly, the diagrams of Figs. 1 and 2 are not normally said to represent equivalent functions. Yet Table III is identical to Table II within a permutation (only the first two columns need be interchanged) and hence from our viewpoint the functions of these two tables *are* equivalent.

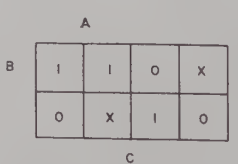


Fig. 1—Veitch diagram for Tables I and II.

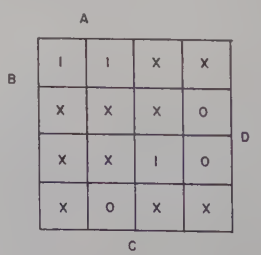


Fig. 2—Veitch diagram for Table III.

* Received by the PGEC, October 24, 1960; revised manuscript received, May 12, 1961.

† Electronics Lab., General Electric Co., Syracuse, N. Y.

In other words, the problem of synthesizing the function of Table I given that inputs A, B, and C are available is *not* the same as when (as in Table II) the input $\overline{B}$ is also available. (In particular, an inverter would be required in the former case.) However, the problem of synthesizing the function of Table II is exactly the same as that involved with Table III. (Once one function is synthesized, the other may be realized by simply changing the proper inputs.)

Before proceeding with a detailed description of this synthesis procedure, it is well to outline the steps which are to be followed. First, we shall introduce the notion of a *logically passive function.* Roughly speaking, a LPF is one which can be synthesized with only AND and OR gates; no inversion of the input variables is required. A technique will then be described for making any given function logically passive by adding the complements of various input variables as additional columns of the given truth table. The reason for making the function logically passive is twofold: 1) Once the function has been made logically passive, a considerable reduction of its truth table can often be made by removing various rows and columns, and 2) the resulting synthesis procedure for a LPF is much more straightforward than in the general case when inversion of certain input variables may be required.

Having described the procedure for making a function logically passive we shall proceed to techniques for reducing this logically passive truth table. First, a rule for eliminating variables (columns) will be given, then similar rules for eliminating rows. Finally, a method will be described for synthesizing this reduced truth table in a two-level irredundant form.

The remainder of the paper will examine various properties of LPF including a method for synthesis with 3-input majority gates.

II. Notation and Basic Definitions

By a switching function $F(x_1, x_2, \cdots, x_n)$, we shall mean an $(n+1)$-column truth table defining for each n-bit binary input combination the corresponding value of F (either 0 or 1). No assumption of logical independence between columns will be made nor will any condition be imposed on the number of rows. The only requirement will be that the table be *consistent, i.e.,* no n-bit input combination shall appear twice—once with $F=0$ and once with $F=1$.

The n-bit input combinations which appear as rows in the table will be referred to as *assignments* and denoted by α's and β's. An assignment α_1 will be said to *imply* a second assignment α_2 if and only if α_2 has 1's wherever α_1 does. This relationship will be denoted as $\alpha_1 \leq \alpha_2$.[1] Thus, $(10100) \leq (10110)$, $(100) \leq (100)$, $(0) \leq (1)$, etc.

[1] Clearly, the α's form a partially ordered set under this relationship.

Now we shall define a *logically passive function.* All of the results in this paper rest on this concept.

Definition 1—A function $F(x_1, x_2, \cdots, x_n)$ will be said to be an LPF with respect to $(x_1, x_2, \cdots, x_n)$ if and only if for any two assignments α_1 and α_2 if $\alpha_1 \leq \alpha_2$ then $F(\alpha_1) \leq F(\alpha_2)$.[2]

For example, consider the function defined by the following table:

A	B	C	F
1	1	0	0
0	1	0	1
1	0	1	1

F is not logically passive with respect to A, B, C because $(010) \leq (110)$ yet $F(010) \not\leq F(110)$. However, if $\overline{A}$ were also available as an input then the table would become:

$\overline{A}$	A	B	C	F
0	1	1	0	0
1	0	1	0	1
0	1	0	1	1

and F would be logically passive with respect to A, $\overline{A}$, B, and C. Thus, it is seen that the *passivity* of F depends directly on the variables available as inputs.

Now a fundamental property of LPF's may be proved:

Theorem 1—Given a function $F(x_1, x_2, \cdots, x_n)$, F is an LPF of $(x_1, x_2, \cdots, x_n)$ if and only if F can be synthesized using only AND and OR gates.

Proof

Sufficiency—It must be shown that if F can be synthesized using only AND and OR gates, then F must be an LPF with respect to its inputs. Assume F was not an LPF. Then, by Definition 1, it would be possible to find two assignments α_1 and α_2 such that $\alpha_1 \leq \alpha_2$ but with $F(\alpha_1)=1$ and $F(\alpha_2)=0$. Assume two such assignments are found and that the inputs to the circuit for F are divided into 3 classes: those which receive 0 under both α_1 and α_2 will be called a-inputs, those which receive a 0 under α_1 and a 1 under α_2 will be called b-inputs, and those which receive a 1 under both will be called c-inputs. (See Fig. 3.) (Note that no input can receive a 1 under α_1 and a 0 under α_2 since, by hypothesis, $\alpha_1 \leq \alpha_2$.) Now assume all the a-inputs are tied to 0, the c-inputs are tied to 1, and the b-inputs are tied together as a single input. The result is a single input, single output circuit which performs logical inversion. But, by definition, F was built using only AND and OR gates. Hence, such a result is impossible and our original as-

[2] For a comparison of LPF's with frontal and unate functions see the Appendix. All frontal and unate functions are logically passive. In fact, both Gilbert [7] and McNaughton [8] show explicitly that these functions have the property required by Definition 1.

sumption that the condition did not hold must have been false. (Strictly speaking, it should also be shown that AND and OR gates alone cannot be used to perform inversion. However, this will be left as an exercise for the reader.)

Necessity—Here we must show that if F is an LPF, then we can synthesize F using only AND and OR gates. Consider all assignments (α_i) for which $F(\alpha_i)=1$. For each assignment α_i construct an AND gate having as inputs those variables which are 1 under α_i. Put the outputs of these AND gates into a single OR gate. (See Fig. 4.) Now it is clear that we have a circuit which

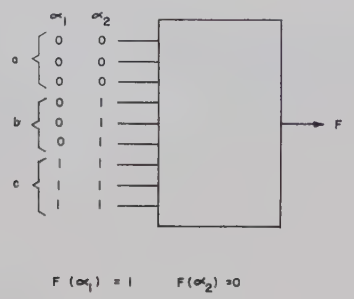


Fig. 3—Reduction of circuit to a single inverter.

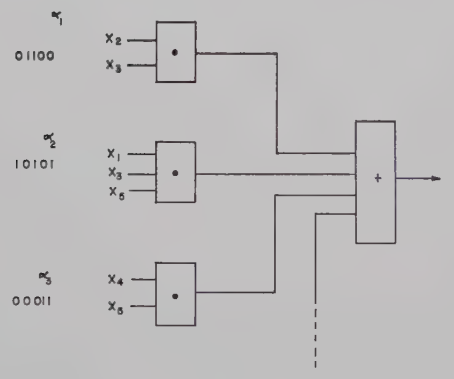


Fig. 4—Two-level synthesis of an LPF.

will have an output of 1 for each α_i. What remains is to consider the assignments (β_j) where $F(\beta_j)=0$. Will the circuit have an output of 0 in these cases? Assume for some β_j it did not. Then, at least one AND gate would have an output of 1, thus implying that all of its inputs were 1 under β_j. But if α_i is the assignment which led to the construction of this gate, this means that $\alpha_i \leq \beta_j$ and, since by definition $F(\alpha_i)=1$ and $F(\beta_j)=0$, this violates the condition of Definition 1. Thus, each β_j must give an output of 0.

(An analogous construction consisting of a bank of OR gates driving a single AND gate could have been obtained by examining the 0's in each β_j.) This synthesis procedure will be referred to frequently in the rest of this paper.

III. Making a Function Logically Passive

Definition 1 provides a simple test for determining whether a given function is logically passive. Moreover, in the case of *active* functions this same technique can be used to determine which variables need to be complemented in order to make the function logically passive.[3] Consider the function described by Table IV:

TABLE IV

A	B	C	F
1	1	0	1
1	0	1	0
0	0	1	1
0	1	0	0

Is F logically passive with respect to A, B, and C? And, if not, what variable, or variables, must be complemented in order to make F logically passive? Applying Definition 1, it is seen that the condition is violated by the second and third rows of the table, *i.e.*, $(001) \leq (101)$ but $F(001) \not\leq F(101)$. Hence we conclude that F is not an LPF with respect to A, B, C.[4] This situation can be remedied by adding a fourth column to the table which will have a 0 in the second row and a 1 in the third. The third assignment will then no longer imply the second and the condition of Definition 1 will not be violated. Obviously, the column (variable) to add is $\overline{A}$ yielding the following Table V:[5]

TABLE V

$\overline{A}$	A	B	C	F
0	1	1	0	1
0	1	0	1	0
1	0	0	1	1
1	0	1	0	0

Now the condition of Definition 1 is satisfied and F is an LPF of $\overline{A}$, A, B, and C. Note that whenever the condition of Definition 1 is violated, there will be at least one variable whose complement may be added to the table to resolve the violation. This follows from the fact that if there were not such a variable then the two assignments under consideration would have to be identical, yet the corresponding values of F would be different. This would mean that the given table was logically inconsistent thus violating our original hypothesis.

[3] A different procedure for finding those input variables which appear complemented in the two-level form is given in Caldwell [1].

[4] Perhaps an easier way to remember the condition of Definition 1 is: Given any two rows of the truth table for F, where F is 0 in one row and 1 in the other, there must exist a column which has a 0 and a 1 in the corresponding rows.

[5] If some other function of A, B, and C had been available as an input and had a 0 in row 2 and a 1 in row 3 this would obviously have worked as well as $\overline{A}$.

It is worth noting that by use of the technique employed in the latter half of the proof of Theorem 1, we can synthesize F directly from Table V. Thus, by noting the units in the rows where F is 1, we obtain

$$F = AB \vee \bar{A}C,$$

or by a similar inspection of 0's where F is 0:

$$F = (A \vee C)(\bar{A} \vee B).$$

In terms of a Veitch diagram, the function of Table V is shown in Fig. 5. Note that the complications which often arise when synthesizing a function having *don't care* conditions were avoided by this *passive function* approach. In fact, all of the methods of this paper are actually made simpler with *don't care* conditions since this means fewer rows in the truth table under consideration.

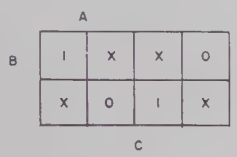


Fig. 5—Veitch diagram for Table V.

In summary: to make a given function logically passive, consider any case where the condition of Definition 1 is violated, *i.e.*, any α_1 and α_2 where $\alpha_1 \leq \alpha_2$, $F(\alpha_1) = 1$, and $F(\alpha_2) = 0$. There will exist an x_i having a 0 in α_1 and a 1 in α_2. (Otherwise the given table will be inconsistent.) Add $\bar{x}_i$ as an additional column. Continue in this manner until all such violations have been resolved.

IV. Eliminating Variables

It is well known that if eliminating a variable from any given truth table results in a table which is still consistent, then any circuit realization of this reduced table will also be a realization of the original table. It is easily seen that this same property holds for LPF's, *i.e.*, if eliminating a variable from the table for an LPF results in a table which is also logically passive then a circuit realization of this reduced table will also be a realization of the original table. To see this we have only to note that if a table is logically passive then from the definition it must be consistent.

Having just examined the problem of adding additional input variables (columns) in order to make a function logically passive, it is natural to ask if a similar procedure will allow certain variables to be eliminated. Consider the function in Table VI.

Application of Definition 1 shows that F is not passive with respect to A, B, and C because of the implication between (001) and (011). However, if $\bar{B}$ is added, this violation is removed as shown in Table VII.

Now F is an LPF with respect to $\bar{B}$, A, B, and C. However, would it still be logically passive if one or more of the variables were eliminated from the table? Obviously, if the elimination of a variable yields a table which still meets the condition of Definition 1, then F is still an LPF and hence can be passively synthesized without the eliminated variable.[6] Thus, in Table VII, the input variable, B, may be eliminated without disturbing the passivity of F.

TABLE VI

A	B	C	F
0	0	0	0
1	1	0	1
0	0	1	1
0	1	1	0

TABLE VII

$\bar{B}$	A	B	C	F
1	0	0	0	0
0	1	1	0	1
1	0	0	1	1
0	0	1	1	0

TABLE VIII

$\bar{B}$	A	C	F
1	0	0	0
0	1	0	1
1	0	1	1
0	0	1	0

Now, we may again synthesize F directly from Table VIII as

$$F = A \vee \bar{B}C,$$

or, in the product of sums form, as

$$F = (A \vee C)(A \vee \bar{B}).$$

The Veitch diagram corresponding to this function is shown in Fig. 6. Note that in synthesizing F all of the *don't cares* were assigned 0's or 1's without explicit examination.

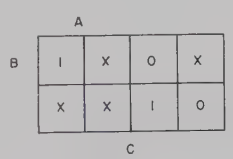


Fig. 6—Veitch diagram for Table VI.

[6] A more formal procedure for eliminating variables is given by Theorem 3 in Section VII.

V. Truth Table Reduction

Not only can we use the fact that a function is logically passive to eliminate columns from its truth table, we can also eliminate many of the rows. Consider the following function in Table IX. It is easily confirmed

TABLE IX

A	B	C	F
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

that F is an LPF of A, B, C. Now consider the first two rows of the table. These state that $F(000)=0$ and $F(100)=0$. But do we really need the information that $F(000)$ must be 0? We know that F is an LPF and since $(000)\leq(100)$ it follows that since $F(100)=0$ then $F(000)$ *has to be 0* in order for F to be logically passive. Therefore, knowing that F is an LPF we would be perfectly justified in eliminating the first row from Table IX. By similar reasoning we note that since $F(011)=F(111)=1$ and $(011)\leq(111)$ we may eliminate (111) from the table. These rules for eliminating rows may be expressed as follows:

Theorem 2—Given a function $F(x_1, x_2, \cdots, x_n)$, where F is an LPF with respect to $(x_1, x_2, \cdots, x_n)$ for any two assignments α_1 and α_2 where $\alpha_1\leq\alpha_2$, if $F(\alpha_1)=F(\alpha_2)=0$ then α_1 may be removed from the table and if $F(\alpha_1)=F(\alpha_2)=1$ then α_2 may be removed.

Applying Theorem 2 to Table IX we see that the first, second, fifth, and eighth rows may be eliminated leaving Table X.

TABLE X

A	B	C	F
0	1	0	0
1	1	0	1
1	0	1	0
0	1	1	1

Again F may be synthesized directly as

$$F = AB \vee BC$$

or as

$$F = B(A \vee C).$$

This property of being able to eliminate rows from the truth table is an important feature of logically passive functions. Roughly speaking, this means that the number of rows (and hence the number of bits) in the truth table of an LPF depends not on the number of variables involved but rather on the relative *complexity* of the function itself. Thus, a *simple* LPF of 15 variables will have even fewer bits in its reduced truth table than many functions of 4 or 5 variables. Particularly with the problem of synthesizing functions, this means that the number of bits to be considered corresponds closely to the number of logical elements required for the function being synthesized.

VI. Logically Passive Functions and Partially Ordered Sets

As has already been observed, the rows of the table of a switching function form a partially ordered set under the binary relation $\leq$. In particular, this means that they may be represented by a Haase diagram. A better insight into the foregoing can be obtained by considering these diagrams. In this type of diagram the elements of the partially ordered set (in this case the binary assignments) are represented as points of a linear graph with the property that if $\alpha_1\leq\alpha_2$ then 1) α_2 appears on a higher level than α_1 and 2) α_1 is connected to α_2 by ascending lines. Consider the function of Table XI. Under the $\leq$ relation these 14 four-bit assignments form the partially ordered set shown in the diagram of Fig. 7. In such a diagram one element is said to *cover* a second if it is on

TABLE XI

A	B	C	D	F
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	1
0	0	1	0	0
1	0	1	0	1
1	1	1	0	1
0	0	0	1	1
1	0	0	1	1
0	1	0	1	0
1	1	0	1	1
1	0	1	1	1
0	1	1	1	0
1	1	1	1	1

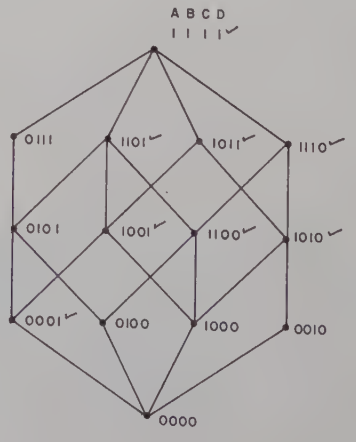


Fig. 7—Haase diagram for Table XI.

a higher level and connected to the second by a single line. Thus (1111) covers (0111), (1001) covers (1000), etc. Now if we define the function of Table XI by indicating by checks those assignments for which F is to be 1 then Definition 1 may be restated as: F is an LPF with respect to its inputs if and only if on the Haase diagram of its assignments no unchecked assignment appears above a checked assignment.

Inspection of Fig. 7 shows that F is *not* an LPF with respect to the given inputs since (0101) appears above (0001). To remedy this condition it is necessary to introduce a fifth input so that this particular implication relation will no longer hold. Obviously the input which will accomplish this is $\overline{B}$. The resulting diagram is shown in Fig. 8. With this additional input, certain of the lines of Fig. 7 disappear; in particular the one between (00011) and (01010). Hence, F is now an LPF.

In terms of this new diagram, Theorem 2 states that since F is an LPF we may remove any checked assignment which appears *above* another checked assignment and likewise for any unchecked assignment which appears *below* another unchecked assignment. Thus, we are left with the diagram of Fig. 9. By examining the checked and unchecked assignments, respectively, we obtain the two two-level forms of F as

$$F = AB \vee A\overline{B}C \vee \overline{B}D$$

and

$$F = (A + \overline{B})(A + B + D)(B + C + D).$$

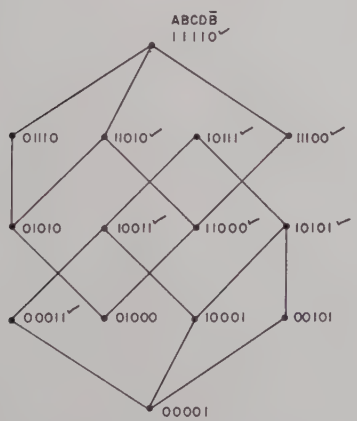


Fig. 8—Haase diagram with $\overline{B}$ as additional input.

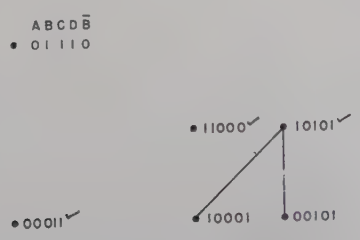


Fig. 9—Reduced Haase diagram.

VII. Synthesis of Logically Passive Functions

The careful reader will have noted that neither of the forms just given for the function of Table XI is irredundant. Thus, the sum of products form can be written as

$$F = AB \vee AC \vee \overline{B}D$$

and the product of sums form as

$$F = (A + \overline{B})(A + D)(B + C + D).$$

We conclude, therefore, that some additional techniques will be required if such redundant forms are to be avoided. It is worth noting, however, that the synthesis procedure already described (in proof of Theorem 1) will often result in an extremely *simple* form for the function if not the *simplest*. As such, it affords a quick method for obtaining a realistic upper bound on the number of logical elements which will be required to synthesize the given function. In the case just considered for example, it was shown that Table XI reduced to that of Table XII. From the rules for synthesizing F, it follows that an upper limit on the number of *gates* in the sum of products form will be one more than the number of rows where F is 1, in this case, four gates. Similarly the number of diode-inputs will be precisely the number of units in these rows (including those in the F-column), in this

TABLE XII

A	B	C	D	$\overline{B}$	F
1	1	0	0	0	1
1	0	1	0	1	1
0	0	0	1	1	1
0	1	1	1	0	0
1	0	0	0	1	0
0	0	1	0	1	0

case, ten. Thus, we know immediately that F can be synthesized in the sum of products form with *at most* four gates and ten diodes. (Actually, as we have seen, four gates and nine diodes are required.) Likewise by inspecting zeros in the rows where F is zero, we see that the product of sums form also requires at most four gates and ten diodes.

Now consider what additional steps must be taken in order to obtain an *irredundant* two-level form for F. We shall use irredundant in the sense of Quine [2], *i.e.*, a form will be said to be irredundant if, in its two-level realization, no input lead may be replaced by a constant without altering the value of the function.

Only the problem of finding the irredundant sum of products form will be considered since the dual procedure for the product of sum form will then be obvious. Consider again the LPF of Table XII. The second row of this table requires that for the input (10101) the output F must be 1. If F is to be constructed in a sum of products form, this means that there must be at least

one AND-gate with only the inputs, A, C, and $\overline{B}$ *or some subset thereof.* This is the heart of the problem. Which inputs must be there and which can be omitted? Once again, Definition 1 affords a simple test for finding which inputs are required.

Suppose in this case A did not appear as an input to this AND-gate. Then F would be 1 not only for (10101) but also for (00101). But note that (00101) already appears in the table with the requirement that F be 0 for this assignment. Hence, to avoid this contradiction A *must* appear as an input to the AND-gate corresponding to (10101). Let us denote this fact by circling the A-unit in the second row and call such a unit an *essential unit.* Likewise, the A-unit in row 1 is essential since, if it were made a 0, F would be 1 for (01000), and yet according to the table F is 0 for (01110). And since (01000) $\leq$(01110) this would violate the condition of Definition 1.

In summary:

Definition 2—Given an assignment α_i where $F(\alpha_i)=1$, a unit in α_i is an *essential unit* (and hence is circled) if and only if when this unit is replaced by a 0 the resulting assignment implies some other assignment β_j where $F(\beta_j)=0$.

Thus, we see that for Table XII all but one of the units in the three rows where F is to be 1 are *essential.* (See Table XIII.)This procedure of finding *essential*

TABLE XIII

A	B	C	D	$\overline{B}$	F
①	①	0	0	0	1
①	0	①	0	1	1
0	0	0	①	①	1
0	1	1	1	0	0
1	0	0	0	1	0
0	0	1	0	1	0

units also provides a check on those variables which can be eliminated since:

Theorem 3—Given an LPF $F(x_1, x_2, \cdots, x_n)$, a variable x_i can be eliminated if and only if its column contains no essential units.

Proof

From the definition of an *essential unit* it follows that given an assignment α_1 to an LPF, F where $F(\alpha_1)=1$, a unit in column x_i of α_1 is essential if and only if there exists an α_2 such that) $F(\alpha_2)=0$ and 2) x_i is the *only* variable with a one in α_1 and a 0 in α_2. But this is precisely the condition (as discussed in Section IV) under which x_i cannot be eliminated.

Returning to the example, application of Theorem 3 to Table XIII shows that none of the variables can be eliminated. Now for any assignment α_i where $F(\alpha_i)=1$ let α_i* denote the assignment obtained by making 0 all units in α_i which are not *essential.* Thus, for Table XIII the three α*'s would be (11000), (10100), and (00011), respectively. It then follows directly that for any α_i* if there does not exist a β_j where $F(\beta_j)=0$ such that $\alpha_i \leq \beta_j$ then the product defined by the units in α_i* must appear in the irredundant sum of products form for F.[7] Such products will be called *essential products.* In this case none of the α_i*'s imply any of the assignments where F is to be 0, so that the irredundant form for F may be read directly from the α_i*'s as,

$$F = AB \vee AC \vee \overline{B}D.$$

Note that when there is only one irredundant form for F, this procedure will automatically find it. However, if there is more than one irredundant form, one additional step must be included in order to reduce F to one of these irredundant forms.[8] This final step consists of selecting any unit which is uncircled in an α_i where $F(\alpha_i)=1$ and changing it to a 0. Essential units are now recalculated (all the old ones will remain, but some new ones may appear in the α_i where the unit was changed to a 0) and the process proceeds as before.

Let us now summarize these steps and illustrate them with two examples from the literature. (We will denote assignments where a given F is to be 1 by α's and assignments where F is to be 0 by β's.) It is understood that the given function is logically passive or has been made logically passive by the procedure of Section III.[9]

1) Reduce table by removing rows. (If $\alpha_i \leq \alpha_j$, remove α_j. If $\beta_i \leq \beta_j$, remove β_i.)
2) Circle all essential units. (A unit in α_i is circled if changing it to a zero would result in α_i implying some β_j.)
3) Reduce table by removing columns. (A column (variable) is eliminated if it contains no essential units. As each column is eliminated Steps 1 and 2 should be repeated. When all columns have essential units proceed to Step 4.)
4) Extract essential products. (The essential units in α_i constitute an essential product if collectively they do not imply any β_j. As each essential product is found, record it, delete its row and all other α_i's which it implies. Return to Step 3. If no essential products are found, proceed with Step 5.)
5) Eliminate an *inessential* unit. (Change an *inessential* unit to zero. An *inessential* unit is an uncircled unit in an α_i. Return to Step 1.)

The process is complete when all α_i's have been removed from the table. To illustrate this procedure, consider the function given by Table XIV.[10] Application of

[7] The products derived of this step constitute what Quine [2] calls the *core* of the function.

[8] In some cases it may be desired to find *all* irredundant forms for F (see Mott [4], for example). This is also possible with this procedure but will not be discussed here.

[9] The quickest way to make a given F into an LPF is to simply include *all* the complements of the variables. The unneeded ones will subsequently be eliminated.

[10] This example is taken from Caldwell [1] where the two-level form is found using Karnaugh maps and by the Quine-McCluskey method [2, 3]. The *don't cares* have been included in Table XIV for convenience in comparing it with these two examples.

Definition 1 shows that $\overline{X}$ must be added. (Specifically, $\overline{X}$ must be included because otherwise $(0010) \leq (0110)$.) In Table XV, $\overline{X}$ has been added, the three *don't cares* deleted, and the rows regrouped as α's and β's.

Now Step 1 is applied with the result that two α's and four β's are deleted leaving Table XVI. Essential units are then circled according to Step 2. Since no units are circled in the X-column, X can be eliminated by Step 3. Application of Step 4 shows that three essential products may be extracted: $\overline{X}Y$, YZ, and WZ. Moreover, the remaining row (10011) is implied by the essential product WZ so it, too, is eliminated and the process ends. Thus,

$$F = \overline{X}Y \vee YZ \vee WZ.$$

TABLE XIV

W	X	Y	Z	F
0	0	0	0	0
0	0	0	1	×
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	×
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	×

TABLE XV

W	X	Y	Z	$\overline{X}$	F
0	0	1	0	1	1
0	0	1	1	1	1
0	1	1	1	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	0	1
0	0	0	0	1	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0

TABLE XVI

W	X	Y	Z	$\overline{X}$	F
0	0	①	0	①	1
0	1	①	①	0	1
1	0	0	①	1	1
①	1	0	①	0	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	0	0	0

In this example there was only one irredundant form for F. Hence, Step 5 was not required. Now consider the the function described in Table XVII.[11] All the complements are needed to make the function logically passive and these with the essential units circled are shown in Table XVIII. Step 1 may be skipped to start since whenever each column appears both complemented and uncomplemented, there can be no implication between rows. All columns have essential units, so we proceed to Step 4. Inspection of Table XVIII reveals two essential products: $\overline{A}B\overline{C}$ and $AB\overline{D}$. When these two rows are removed plus the two other rows implied by these essential products the B column becomes all zeros (in the α's) so that it, too, may be removed. The result is Table XIX.

TABLE XVII

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

TABLE XVIII

A	B	C	D	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$	F
0	0	0	0	1	1	1	①	1
0	0	1	0	①	①	0	1	1
0	0	①	1	1	①	0	0	1
0	1	0	0	1	0	①	1	1
0	①	0	1	①	0	①	0	1
1	0	0	0	0	1	①	1	1
①	0	0	1	0	①	1	0	1
1	0	1	①	0	①	0	0	1
1	1	0	0	0	0	1	①	1
①	①	1	0	0	0	0	①	1
1	0	1	1	0	0	0	0	0
0	0	0	1	1	1	1	0	0
0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0
1	0	1	0	0	1	0	1	0
1	1	0	1	0	0	1	0	0
1	1	1	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0

TABLE XIX

	A	C	D	$\overline{A}$	$\overline{B}$	$\overline{C}$	D	F
	0	0	0	1	1	1	①	1
	0	1	0	①	①	0	1	1
	0	①	1	1	①	0	0	1
Essential	1	0	0	0	1	①	1	1
Products:	①	0	1	0	①	1	0	1
$\overline{A}B\overline{C}$, $AB\overline{D}$	1	1	①	0	①	0	0	1
	0	0	1	1	1	1	0	0
	0	1	0	1	0	0	1	0
	0	1	1	1	0	0	0	0
	1	1	0	0	1	0	1	0
	1	0	1	0	0	1	0	0
	1	1	1	0	0	0	0	0

Step 4 is now repeated but no further essential products are found. Therefore, we proceed to Step 5. Let us select the first *inessential* unit in the A-column, *i.e.*, the first unit of (1000111). This is changed to a zero and we return to Step 1. The first row may now be eliminated since it is implied by the new fourth row. Likewise, at Step 2 the fourth row receives another essential unit in the $\overline{D}$-column. (Note that immediately after Step 5 the only row which can receive new essential units is the one where a unit was changed to a zero.) The result of these steps is Table XX. Step 4 now yields the additional essential product, $\overline{C}\,\overline{D}$. Variables $\overline{C}$ and $\overline{D}$ go out as well as three of the remaining α's and two of the β's. After finding essential units, Table XXI results. Both α's now have essential products, namely: $\overline{A}\,\overline{B}C$ and $A\overline{B}D$. Thus, F is found in irredundant form as:

$$F = \overline{A}B\overline{C} \vee AB\overline{D} \vee \overline{C}\,\overline{D} \vee \overline{A}\,\overline{B}C \vee A\overline{B}D.$$

[11] This example is taken from Phister [5] where the Harvard Chart Method is employed to obtain the irredundant forms.

TABLE XX

A	C	D	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{D}$	F
0	1	0	①	①	0	1	1
0	①	1	1	①	0	0	1
0	0	0	0	1	①	①	1
①	0	1	0	①	1	0	1
1	1	①	0	①	0	0	1
0	0	1	1	1	1	0	0
0	1	0	1	0	0	1	0
0	1	1	1	0	0	0	0
1	1	0	0	1	0	1	0
1	0	1	0	0	1	0	0
1	1	1	0	0	0	0	0

TABLE XXI

A	C	D	$\bar{A}$	$\bar{B}$	F
0	①	0	①	①	1
①	0	①	0	①	1
0	0	1	1	1	0
0	1	1	1	0	0
1	1	0	0	1	0
1	1	1	0	0	0

Essential Products:
$\bar{A}B\bar{C}$, $AB\bar{D}$, $\bar{C}\,\bar{D}$

Happily, this is one of the two irredundant forms given in [5]. Selection of other inessential units at Step 5 will lead to the other irredundant forms. Actually, there are five irredundant, sum-of-products forms for this F.

As mentioned earlier, there is a duality between the steps for finding F in an irredundant sum-of-products form and those for a product-of-sums form. In fact, to find the steps for the latter case the five steps need merely be modified as follows:

Interchange: α and β,
$\leq$ and $\geq$
unit and *zero*,
product and *sum*.

Before leaving this section on synthesis, several comments seem appropriate. The reader's first reaction to the foregoing may well be that the one thing which the general area of switching circuit theory does *not* need is *another* method for synthesizing combinational logic. However, this method does offer several features which may make it more desirable in certain applications:

1) It is completely general with regard to the switching function to be synthesized, *i.e.*, any mapping of binary input combinations onto 0 and 1 can be handled. *Don't care* conditions not only do not cause complications but rather reduce the number of input combinations to be considered.
2) By initially removing rows from the truth table (via Theorem 2) a considerable reduction in the number of bits to be processed can often be achieved.
3) Realistic upper bounds on the number of logical elements required for the sum-of-products and the product-of-sums forms are available immediately from the reduced truth table.
4) The actual synthesis procedure may be carried out completely on the original truth table, *i.e.*, rows and columns are crossed out, essential units are circled, essential products are checked, etc. No auxiliary charts, maps, or prime implicant tables are required.
5) The entire synthesis process involves only one non-trivial operation—that of determining whether one binary number implies another. As such it is especially suited for programming on a digital computer.[12]

It is difficult to make a detailed appraisal of the merit of this procedure relative to such standard methods as Quine-McCluskey [1–3] or Veitch diagrams and Karnaugh maps [1, 5]. For this reason, we have deliberately chosen examples which have been treated elsewhere by these methods. We shall limit our remarks, therefore, to pointing out some advantages and disadvantages of each method.

The Veitch diagram-Karnaugh map approach seems to be definitely superior for problems involving a small number of variables, up to five or six. Not only are these diagrams easy to handle, but also the user gains a valuable insight into the process itself. However, they quickly become unwieldy for greater numbers of variables.

The Quine-McCluskey method is widely used and most other methods rest on its basic concepts. It is especially effective when the total number of assignments for which F is to be 1 (or 0) is small compared to 2^n. One main disadvantage, as with many related methods, lies with the length of the list of prime implicants which must be initially generated and subsequently examined. For large problems the number of entries on this list may well exceed 2^n.

The apparent advantages of the *LPF approach* are enumerated above. At least two disadvantages are: 1) with *complete* switching functions (no *don't cares*) all 2^n rows of the truth table must be specified initially (although hopefully many will subsequently be eliminated). This is in contrast to the Quine-McCluskey method where only the rows for which F is to be 1 (or 0) need be listed. 2) Because it is specifically tailored for use on a digital computer, using this method for hand-computation can become tedious although certainly it is not complicated to apply.

VIII. Some General Properties of Logically Passive Functions

Practically all of the properties and procedures associated with Boolean functions have their counterparts with LPF's. Thus, equations involving LPF's may be solved, methods exist for decomposing LPF's, symmetric LPF's may be identified, etc. Here, however, we shall limit our discussion to a few of the more basic properties.

[12] In this regard, the above procedure (extended to accommodate multiple outputs) is currently being employed as the basic algorithm in the SYLO (SYnthesis of LOgic) program. This program (originally written for the 704 and now rewritten for the 7090) provides automatic synthesis of multiple-output combinational logic using preselected logical elements, *e.g.*, AND-gates, OR-gates, inverters, NOR-gates, majority gates, and subject to various engineering constraints such as fan-in, fan-out and number of levels of logic. Using this procedure, the synthesis in two-level form of a B.C.D. serial adder (9 inputs, 5 outputs) takes approximately one minute. Synthesis of a function such as in Table XVII takes a matter of seconds.

A. Expansion of LPF's

Given a Boolean function $G(x_1, x_2, \cdots, x_n)$, it is well known that G can be expanded about a variable, *e.g.*, x_1 as:

$$G(x_1, x_2, \cdots x_n) = x_1 G(1, x_2 \cdots, x_n) \vee \bar{x}_1 G(0, x_2, \cdots, x_n).$$

By repeated applications of this relation, expansion about any subset of the basic variables may be obtained. Similarly, using the concepts of a *Boolean difference* (see [6]) G can be expanded in a manner analogous to the classical Taylor and Maclaurin series expansions.[13]

In order to investigate the expansion of LPF's, one further definition is required:

Definition 3—Given an LPF, $F(x_1, x_2, \cdots, x_n)$, described in truth table form, ${}^{x_i}F^1$ will denote the function whose truth table is obtained by deleting from the table for F all rows where both F and x_i are 0 and then deleting the column, x_i. Similarly ${}^{x_i}F^0$ will denote the function whose truth table is obtained by deleting all rows where both F and x_i are 1 and then deleting the x_i-column.

Table XXII shows this definition more clearly. Note that both ${}^{x_i}F^1$ and ${}^{x_i}F^0$ will be LPF's. The justification for eliminating the x_i column is obvious since x_i has no 0-1 pairs in common with either ${}^{x_i}F^1$ or with ${}^{x_i}F^0$. Furthermore, it can be seen from the table that F can be expressed as:

$$F = x_i {}^{x_i}F^1 \vee {}^{x_i}F^0 \tag{1}$$

or as

$$F = (x_i \vee {}^{x_i}F^0) {}^{x_i}F^1. \tag{2}$$

TABLE XXII

x_i	F	${}^{x_i}F^1$	${}^{x_i}F^0$
0	0	—	0
1	0	0	0
0	1	1	1
1	1	1	—

Obviously, ${}^{x_i}F^1$ and ${}^{x_i}F^0$ can each be expanded again in terms of a second variable and so on, so that expansion about any set of x_i's can be obtained. If this expansion process is used as a means of synthesizing F, several comments are appropriate: First, at each stage of the process an x_i must be selected about which the function is to be expanded. A good rule for choosing this variable is to select the one with the most essential units and essential zeros in its column. Secondly, if the particular form in which an expanded F is to be synthesized, *i.e.*, (1) or (2), is known, a further simplification can be made. Thus, from Table XXII it is seen that F is to be expressed in the form of (1) then the rows where x_i is 0 and F is 1 can be eliminated from the table for ${}^{x_i}F^1$. Likewise, when using form (2) the rows where x_i is 1 and F is 0 can be removed from the table for ${}^{x_i}F^0$.

[13] A *difference* function can also be defined for LPF's and used to define various of their formal properties.

B. Synthesis of LPF's with Three-Input Majority Gates

With the growing importance of the three-input majority gate as a basic logical element, it is worth noting that LPF's may be synthesized very easily using these gates. Referring again to Table XXII, it is seen that

$$F = M(x_i, {}^{x_i}F^1, {}^{x_i}F^0) \tag{3}$$

where M denotes the three-input majority gate whose output is equal to the majority of its inputs. (Note that the majority function is itself an LPF even though it is not normally built out of AND- and OR-gates.)

To illustrate the application of (3) to the synthesis of a function using only three-input majority gates, consider again the function given in Table XIV. As was shown, when F is made logically passive this table reduces to that of Table XVI. Starting with this table, Fig. 10 shows how F may be successively expanded in terms of three-input majority gates.

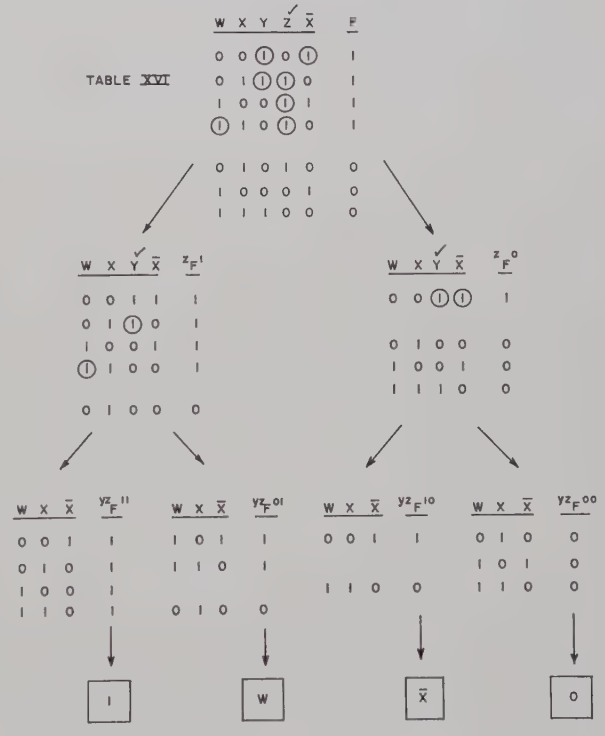


Fig. 10—Expansion of Table XVI.

Thus, F is first expanded about Z. (Here again the number of essential units is used to select the variable for expansion.) ${}^zF^0$ and ${}^zF^1$ are then each expanded about Y where the resulting functions are seen to be 1, W, $\overline{X}$, and 0, respectively. Finally, F is *reassembled* yielding the circuit shown in Fig. 11. (Actually in practice the above procedure is even simpler since numerous rows and columns may be eliminated at each step.)

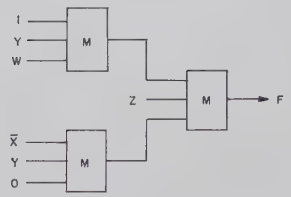


Fig. 11—Synthesis with three-input majority gates.

This synthesis procedure makes no claim to minimality with regard to the number of majority gates required. However, it does have the virtue of being extremely straightforward and as such can form the basis for more sophisticated techniques. Also it has the important feature that it automatically subsumes *don't care* conditions.

C. LPF's and Inverters

The approach to combinational logic described in this paper is analogous to one commonly employed with sequential circuits, *i.e.*, a sequential circuit is often viewed as a collection of blocks of combinational logic interconnected with unit time delays. Here, we have looked on a combinational circuit as a collection of LPF's interconnected with inverters.[14] Thus, if desired, any sequential circuit may be described in terms of LPF's, unit time delays, and inverters.

In the synthesis procedures thus far described, inverters, if they appear at all, will appear only on the inputs to the circuit involved. However, a saving in logical elements will often result if the required inversion can be done at other locations in the circuit. Thus, using the above procedures to synthesize $\overline{A} \vee \overline{B}$ from inputs A and B, the resulting circuit would consist of a two-input OR-gate with an inverter on each input. Obviously, a better realization would be with a two-input AND-gate having a single inverter on its output.

This technique of minimizing the number of inverters rests on the concept of the *dual* of a function. Given an LPF, F, in truth table form, the dual of F, $D(F)$ is defined as that function whose truth table is obtained by complementing every bit in the table for F. Since $a \leq b$ if and only if $\bar{b} \leq \bar{a}$ it follows that $D(F)$ will likewise be logically passive. Thus,

$$D(F) = \overline{F}(\bar{x}_1, \bar{x}_2, \cdots \bar{x}_n)$$

or

$$F(x_1, x_2, \cdots x_n) = \overline{D}[F(\bar{x}_1, \bar{x}_2, \cdots \bar{x}_n)].$$

In other words, any LPF, F, is logically equivalent to the LPF, $D(F)$, with its inputs and its output inverted. Likewise, as is well known, when F is built with AND- and OR-gates, $D(F)$ is obtained by simply interchanging the two types of gates. Some gates, *e.g.*, majority gates, are their own duals so that here inputs and outputs may be inverted without any other change in the circuit. Using these relationships inverters can often be moved *into* a synthesized LPF with a resultant saving in hardware. Note that this conversion process may be applied not only to the entire synthesized circuit, but also to any portion thereof. An interesting problem is to find an algorithm for systematically applying this technique to a given circuit so that the configuration requiring the fewest number of inverters results.

[14] This approach needs to be modified when the basic logical element involves inversion, *e.g.*, NOR-gates, just as with the sequential circuit approach when a basic element involves time delay and combinational logic, *e.g.*, flip-flops.

IX. Conclusions

The general approach to logical switching functions followed in this paper differs considerably from the usual algebraic and topological methods appearing in the literature. By confining our attention only to the truth table of the function we have sacrificed much of the power of the algebraic methods and the neat geometric analogies of the topological methods. On the other hand, these losses appear to be offset by 1) the fact that *all* switching functions are equally amenable to these techniques, and 2) the feature of the extreme simplicity of the underlying methods. Particularly with regard to automation of logical design procedures on a digital computer this approach seems promising.

Appendix

Frontal, Unate, and Logically Passive Functions

Logically passive functions are closely related to the frontal functions of Gilbert [7] and the unate functions discussed by McNaughton [8]. (See Quine [9] also.) However, LPF's differ from frontal and unate functions in several respects: 1) in order to apply the basic definitions of frontal and unate functions an actual realization of the function, in at least an algebraic sense, is required, 2) both frontal and unate functions are defined only for *complete* switching functions, *i.e.*, switching functions with no *don't care* conditions, and 3) with both frontal and unate functions the *labels* which appear on the inputs must be considered. Let us examine these differences.

With frontal functions an actual (contact) network realization of the function is assumed. If all contacts of this network are front contacts, then the switching function which describes this network is said to be a frontal switching function. A unate function is a switching function which can be represented as a normal formula in which no variable appears both negated and unnegated. Note that both of these definitions assume a realization, in a circuit or an algebraic sense, in order to be applied. As such they are completely satisfactory for the author's purposes. However, as a basis for a synthesis procedure a definition which avoids any preliminary realization of the function is preferred.

Both frontal and unate functions are defined only for

complete switching functions although each could possibly be extended to *don't care* conditions. Thus, an *incomplete* switching function could be said to be frontal if *there exists* a contact network realization in which all contacts are front contacts. Likewise, for unate functions, an *incomplete* function could be said to be unate if *there exists* a normal form representation of the function in which no variable appears both negated and unnegated. However, here again, a realization of the function would be required in order to apply these definitions. And, in addition, all possible realizations might have to be considered.

Finally, even if the definitions of frontal and unate were extended to include the *incomplete* functions both types would still differ from LPF's because of their dependence on *labelling*. Consider the following function:

X_1	X_2	X_3	X_4	F
1	0	0	1	0
0	1	0	1	1
1	0	1	0	1
0	1	1	0	0

If we let $X_1 = A$, $X_2 = B$, $X_3 = C$, and $X_4 = D$, then this function can be represented in normal form as $AC \vee BD$ and a corresponding contact network realization exists. As such it is both frontal and unate. However, if we change the X_4 label to $\overline{D}$, then the function is no longer frontal but is still unate. Finally, if we let $X_1 = A$, $X_2 = \overline{A}$, $X_3 = B$, $X_4 = \overline{B}$, then the function is neither frontal nor unate. Clearly the function is logically passive in all three cases since the table itself meets Definition 1 and the actual labelling is not even considered.

In summary, LPF's are a proper subset of all switching functions, unate functions are a proper subset of all LPF's, and frontal functions are a proper subset of all unate functions.

References

[1] S. R. Caldwell, "Switching Circuits and Logical Design," John Wiley and Sons, Inc., New York, N. Y.; 1958.
[2] W. V. Quine, "On cores and prime implicants of truth functions," *Am. Math. Monthly*, vol. 66, pp. 755–760; September, 1953.
[3] E. J. McCluskey, "Minimization of Boolean functions," *Bell Sys. Tech. J.*, vol. 35, pp. 1417–1444; November, 1956.
[4] T. H. Mott, "Determination of the irredundant normal forms of a truth function by iterated consensus of the prime implicants," IRE Trans. on Electronic Computers, vol. EC-9, pp. 245–252; June, 1960.
[5] M. Phister, "Logical Design of Digital Computers," John Wiley and Sons, Inc., New York, N. Y.; 1958.
[6] S. B. Akers, "On a theory of Boolean functions," *J. Soc. Ind. Appl. Math.*, vol. 7, pp. 487–498; December, 1959.
[7] E. N. Gilbert, "Lattice theoretic properties of frontal switching functions," *J. Math. Phys.*, vol. 33, pp. 57–67; April, 1954.
[8] R. McNaughton, "Unate truth functions," IRE Trans. on Electronic Computers, vol. EC-10, pp. 1–6; March, 1961.
[9] W. V. Quine, "Two theorems about truth functions," *Bol. Soc. Math. Mex.*, vol. 10, pp. 64–70; March, 1953

The Use of the Simplex Algorithm in the Mechanization of Boolean Switching Functions by Means of Magnetic Cores*

SIDNEY N. EINHORN†

Summary—**An algorithm is described for mechanizing Boolean switching functions by means of a net of magnetic toroidal cores. The algorithm is referred to as Simplex and, in this application, is programmed for a digital computer. Computer solutions specify the wiring configuration for a core or net of cores yielding a device for performing combinational logic. Switching functions are realizable in essentially one clock time. The logical designer may synthesize a function directly from the truth table without proceeding in the customary manner of expressing the function in Boolean canonical form and then attempting to minimize with respect to hardware or other criteria by means of algebraic manipulation or a mapping or charting technique.**

* Received by the PGEC, November 28, 1960; revised manuscript received, August 14, 1961. The work was supported by the USAF under WADD Contract AF33(616)6355. Reproduction in whole or in part is permitted for any purpose to the U. S. Govt.

† Burroughs Corp., Burroughs Labs., Paoli, Pa.

Introduction

In recent years, many magnetic devices have been introduced which are used to mechanize switching functions. One class of devices which has received wide attention is the multiaperture or multipath structures, such as the Transfluxor,[1] the MAD,[2] and the Laddic.[3] Their common property, in addition to using magnetic material with a rectangular hysteresis loop, is that some or all of the logical functions are performed

[1] J. A. Rajchmann and A. W. Lo, "The Transfluxor," Proc. IRE, vol. 43, pp. 321–338; March, 1956.
[2] H. D. Crane, "A high-speed logic system using magnetic elements and connecting wire only," Proc. IRE, vol. 47, pp. 63–73; January, 1959.
[3] U. F. Gianola, "The Laddic—a magnetic device for performing logic," *Bell Sys. Tech. J.*, vol. 38, pp. 45–72; January, 1959.

by controlling the actual switching path through the structure. The philosophy is to achieve as much logic as possible within the structure thereby reducing component count and interwiring.

The question arises, however, as to whether the simple toroidal core has been given sufficient consideration. A study of Karnaugh's paper[4] and an internal report by Minnick[5] revealed extremely interesting logical properties of the simple toroidal core based on the exploitation of its magnetic threshold. For example, the logical threshold function can be mechanized in a single core including, of course, the AND and the OR functions which are special cases of the threshold function. This leads directly to the consideration of methods of synthesizing any arbitrary Boolean function with one or more cores.

The method developed for achieving this goal utilizes a linear programing technique referred to as the Simplex algorithm. The computer solution indicates the wiring configuration for the core or net of cores. The resulting device realizes a function of n variables in one clock time.

Of extreme importance is the fact that the logical designer may synthesize a function directly from the truth table without proceeding in the customary manner of expressing the function in Boolean canonical form and then attempting to minimize the hardware by algebraic manipulation or some other charting or mapping technique. It is interesting to note that this technique is not limited to magnetic cores but should apply to any device with a physical threshold.

Threshold Function Synthesis with Cores

The magnetic logic discussed in this paper is performed with a two-phase clock. The core is initially in the reset state. At phase one time, the core is set or not set in accordance with the logical requirements. At phase two time, the core is reset. Signals induced by flux change during reset are interpreted as logical ONEs.

The schematic representation of magnetic circuits is facilitated by the use of mirror symbols.[4] In this paper, cores (toroids) are represented by heavy vertical line segments, wire leads by horizontal line segments, and windings by 45° mirror symbols at the intersections of the cores and leads (Fig. 1). The sense of the magnetic field associated with a current in a given winding is obtained by "reflecting" the current in the winding mirror symbol. For example, if the core is initially reset, application of current I_1 to winding N_1 is in a direction to cause the core to set. If the core were initially set, the application of I_2 through N_2 would cause the core to reset.

[4] M. Karnaugh, "Pulse switching circuits using magnetic cores," Proc. IRE, vol. 43, pp. 570–584; May, 1955.
[5] R. C. Minnick, "Linear Input Logic," Burroughs Corp., Paoli, Pa., Internal Rept.; October, 1958.

The threshold function is a Boolean function which states that if μ or more of n variables are equal to ONE, then an output will result. For example, the truth table for $\mu=2$ and $n=3$ is shown in Fig. 2. This function may be mechanized with a single core. It is assumed that a unit current pulse is available to represent each variable and its prime. Karnaugh's[4] rule for generating the threshold function is to connect $(\mu-1)$ of the primed variables in a direction to reset the core and $(n+1-\mu)$ unprimed variables in a direction to set the core. One way to wire the core is shown in Fig. 3. The mmf equation is

$$F = I_2N_2 + I_3N_3 - I_1'N_1. \tag{1}$$

Assuming that the magnetic threshold of the core is one ampere-turn, and noting that all quantities in the mmf

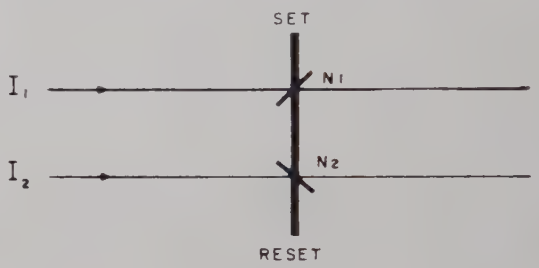


Fig. 1—Mirror symbol notation for core winding.

	I_3	I_2	I_1	f
i=0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Fig. 2—Truth table for threshold function where $\mu=2$ and $n=3$.

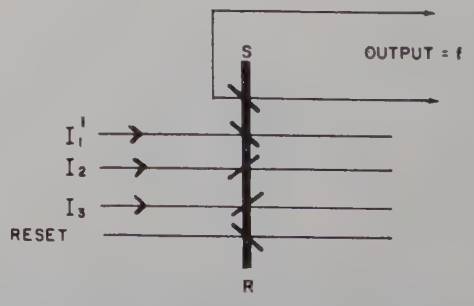


Fig. 3—Core and wiring configuration for threshold function where $\mu=2$ and $n=3$.

equation are integers, it may be stated that

$$\text{if } F_i \geq 1, \qquad \text{then } f_i = 1 \tag{2a}$$

and

$$\text{if } F_i \leq 0, \qquad \text{then } f_i = 0 \tag{2b}$$

where the index "i" denotes the row of the truth table, and F_i and f_i the corresponding values of the mmf and Boolean function, respectively. It is readily seen that if $N_1 = N_2 = N_3 = 1$, and if the values of the variables are applied to (1) in accordance with the truth table, the proper values of F_i are attained to satisfy the threshold function required.

Extension to Arbitrary Functions

An extension to this type of switching function mechanization is a technique for synthesizing any arbitrary Boolean function in a similar manner. Many functions can be mechanized with a single toroid, but some require more than one. For example, the Exclusive-OR function requires two cores. The method to be described accommodates both single and multicore functions.

Consider the function 01011101 (where 1 denotes the truth table entry for $i = 0$). The truth table representation is shown in Fig. 4. The function is mechanized in a single core if the core is wound according to the equation,

$$F = 2I_0 - 2I_1 + I_2 - I_3. \tag{3}$$

(Currents I_0 and I_2 are applied to two turns and one turn, respectively, in a direction to set the core, and I_1 and I_3 are applied to two turns and one turn, respectively, in a direction to reset the core; current I_0 is a pulse bias current always present.) This solution may be reached by inspection or by trial and error. When the number of variables increases, the need for an algorithm becomes apparent.

	I_0	I_3	I_2	I_1	f
i =0	1	0	0	0	1
1	1	0	0	1	0
2	1	0	1	0	1
3	1	0	1	1	1
4	1	1	0	0	1
5	1	1	0	1	0
6	1	1	1	0	1
7	1	1	1	1	0

Fig. 4—Truth table for $f = 01011101$.

For each row of the truth table of Fig. 4, an mmf inequality may be written. A threshold value of one ampere-turn is assumed for core switching and, for conditions where core switching is not desired, the net mmf is held less than or equal to zero ampere-turns. The inequalities are as follows:

$$\begin{aligned}
&I_0N_0 && && && \geq 1\\
&I_0N_0 + I_1N_1 && && && \leq 0\\
&I_0N_0 && + I_2N_2 && && \geq 1\\
&I_0N_0 + I_1N_1 && + I_2N_2 && && \geq 1\\
&I_0N_0 && && + I_3N_3 && \geq 1\\
&I_0N_0 + I_1N_1 && && + I_3N_3 && \leq 0\\
&I_0N_0 && + I_2N_2 && + I_3N_3 && \geq 1\\
&I_0N_0 + I_1N_1 && + I_2N_2 && + I_3N_3 && \leq 0.
\end{aligned} \tag{4}$$

The system of inequalities (4) is consistent (one or more solutions), and the function may be mechanized by a single core. It inequalities (4) were inconsistent, then more than one core would be required.

It will be shown how the above inequalities are processed by Simplex. If the system of inequalities is consistent, a set of values for N_j will be found which satisfies the inequalities. However, it is important to note that a particular solution will be found which will minimize an arbitrarily specified linear function, referred to as the *objective* or *cost* function of the form

$$Z = K_1N_1 + K_2N_2 + \cdots + K_nN_n \tag{5}$$

where $K_1, K_2, \cdots, K_n$ are constants. The method of handling inconsistent sets of inequalities (multicore functions) by Simplex will also be demonstrated.

Dummy Variables in Multicore Solutions

Before proceeding with the explanation of the Simplex algorithm, consider the single-core and two-core examples of solutions of two functions. The function 01011101 has the single-core solution $N_0 = 2$, $N_1 = -2$, $N_2 = 1$, and $N_3 = -1$ as represented by (3). Now consider the function $h = 0110001\hat{0}$. The Simplex-derived solution is $N_1 = 1$, $N_2 = -1$, and $d_6 = K$. If $d_6 = K$ were not in the solution, the mmf equation would be written as

$$F = I_1 - I_2. \tag{6}$$

A core wound in accordance with (6) yields the function $f = 00100010$. Note that f differs from h by its absence of a ONE in the sixth-bit position. This corresponds to the subscript of d_6, a *dummy variable*, which plays a crucial role in multicore solutions.

A new function g is formed by observing the dummy variables remaining in the solution, and placing a ONE in the bit positions corresponding to the subscripts of the dummy variables. In this example, $g = 01000000$. This function is now processed and the solution is

$N_0=-1$, $N_1=-1$, $N_2=1$, $N_3=1$. The corresponding mmf equation is

$$G = -I_0 - I_1 + I_2 + I_3. \tag{7}$$

Since no dummy variables remain in the solution, g is a single-core function. In general, g might not be a single-core function, in which case the procedure described above would be repeated until the final function processed contained no dummy variables. In our example, two cores are required to mechanize h (Fig. 5). Cores F and G are wound according to (6) and (7), respectively. An output wire is series-connected to both cores. Either core F or core G, or neither, will switch in accordance with the input lines energized. Hence, the voltage induced in the output line at phase two time represents the function h, or

$$h_i = f_i + g_1. \tag{8}$$

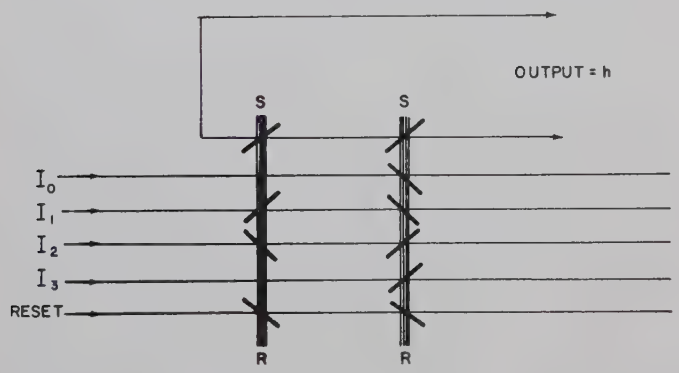


Fig. 5—Two-core mechanization of $h=01100010$.

Explanation of Simplex

It is the aim of this paper to provide an insight into the Simplex method and its application to our problem while avoiding the many details associated with computation and programing. The Simplex algorithm is perhaps more easily understood if at first a geometrical interpretation is given. Consider the following system of inequalities:

$$\begin{aligned}
&(1)\ a_{11}x_1 + a_{12}x_2 \leq b_1\\
&(2)\ a_{21}x_1 + a_{22}x_2 \leq b_2\\
&(3)\ a_{31}x_1 + a_{32}x_2 \geq b_3\\
&(4)\ a_{41}x_1 + a_{42}x_2 \leq b_4.
\end{aligned} \tag{9}$$

If each of the inequalities (9) is treated as an equation and is graphed, the set of lines shown in Fig. 6 would result. Only the area included by the convex polygon *abcd* contains values of x_1 and x_2 which are solutions to equalities (9).

A solution which minimizes an associated cost function of the form $Z=AX_1+BX_2+C$ is readily obtained from the following useful theorem.

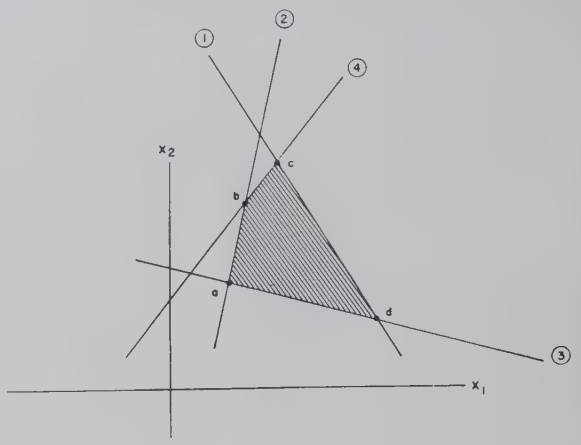


Fig. 6—Convex polygon of (9).

Theorem I

A linear function defined over a convex polygon takes on its maximum and minimum values at corner points of the convex polygon.[6]

Hence, the value of the cost function is simply tested at each corner of the convex polygon to determine the values of x_1 and x_2 which satisfy the system of inequalities and minimize the cost function. As the number of variables and equations increases, the geometrical picture becomes hopeless. It would amount to finding corner points of convex hulls in hyperspace. Fortunately, the use of the Simplex algorithm does not depend upon the geometrical picture. The algorithm is best explained by example, accompanied by pertinent theorems and definitions.

Consider the following system of inequalities and cost function.

$$\begin{aligned}
&(1)\ x_1 \geq 0\\
&(2)\ x_2 \geq 0\\
&(3)\ x_1 + 2x_2 \leq 8\\
&(4)\ x_1 + x_2 \leq 5\\
&(5)\ -x_1 + 2x_2 \leq 6\\
&Z = -2x_1 - 3x_2.
\end{aligned} \tag{10}$$

The convex polygon is shown in Fig. 7. The coordinates of the corner point (2, 3) minimize the cost function as can be checked by substitution.

The Simplex algorithm is utilized to find what is referred to as a basic feasible solution:

Definition I: A feasible solution is a solution which contains non-negative x's.

Definition II: A basic feasible solution is a solution which contains at most m positive x's, where m is the number of constraints.

[6] J. G. Kennedy, *et al.*, "Introduction to Finite Mathematics," Prentice-Hall, Inc., Englewood Cliffs, N. J., pp. 256–257; 1956.

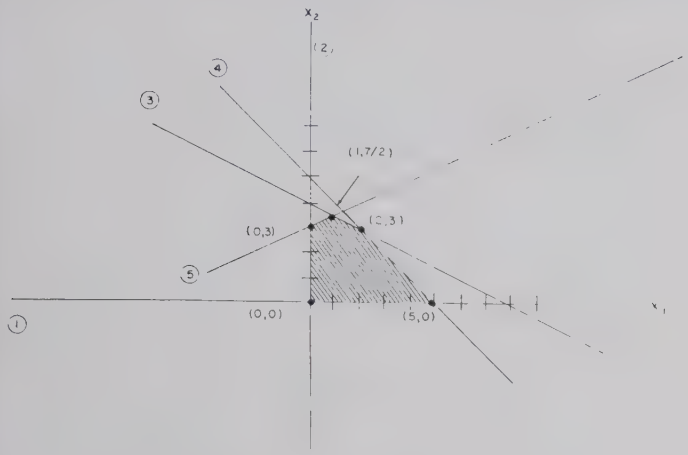


Fig. 7—Corner points of the convex polygon of (10).

Theorem II

If a problem has an optimum feasible solution, then it also has a basic feasible solution which is optimum.[7]

Since the solutions are restricted to basic feasible solutions, the first two inequalities of (10) need not be written. The last three inequalities of (10) are converted to equations by the introduction of *slack* variables x_3, x_4, and x_5, as follows:

$$\begin{aligned} x_1 + 2x_2 + x_3 &= 8 \\ x_1 + x_2 + x_4 &= 5 \\ -x_1 + 2x_2 + x_5 &= 6 \\ Z &= -2x_1 - 3x_2. \end{aligned} \tag{11}$$

Eq. (11) is now said to be in "canonical" form with respect to variables x_3, x_4, and x_5. The canonical form requires that the coefficient of x_3 is plus one, and that x_3 does not appear in any other equation. This is true also for x_4 and x_5.

Note that the canonical form of (11) enables one to obtain a basic feasible solution simply by setting x_1 and x_2 equal to zero. One solution, then, is $x_3=8$, $x_4=5$, $x_5=6$ and the corresponding value of the cost function $Z=0$. The introduction of slack variables provides an *initial* basic feasible solution not necessarily optimum. *By appropriate algebraic manipulation, (11) can be put into canonical form with respect to any set of three x's.* These variables are referred to as *basic variables* or as the *basis*. In this problem, the total number of solutions to be examined is the combination of five things, taken three at a time, or ten solutions. For larger problems this trial and error procedure becomes prohibitive. Fortunately, there is a systematic iterative procedure for finding, in a relatively small number of steps, a solution which minimizes the cost function.

[7] R. Dorfman, *et al.*, "Linear Programming and Economic Analysis," McGraw-Hill Book Co., Inc., New York, N. Y., p. 78; 1958.

Manipulation of Equations

With reference to the cost function of (11), if x_1 or x_2 were increased from zero, then the value of Z would be reduced. This, of course, is because their coefficients, called the *relative cost factors*, are negative. First to be considered is x_2, since it would appear that any change here would have the greater effect on the value of Z. (It is not essential, however, that x_2 be given first consideration.) The question arises as to how large x_2 can be made without causing any of the other x's to turn negative. Therefore, setting all other variables equal to zero in (11),

$$\begin{aligned} 2x_2 = 8; &\quad \therefore\ x_2 = 4 \\ x_2 = 5; &\quad \therefore\ x_2 = 5 \\ 2x_2 = 6; &\quad \therefore\ x_2 = 3. \end{aligned} \tag{12}$$

The minimum positive value of x_2 is selected; *i.e.*, $x_2=3$. If the value of 4 were selected, then x_5 would turn negative. If the value of 5 were selected, then both x_3 and x_5 would turn negative. The value of $x_2=3$ makes $x_5=0$. An expression for x_2 in terms of x_5 is then found from the third equation and substituted in the others, including the cost function. Thus, x_2 replaces x_5 in the basis, and (11) is now in canonical form with respect to the basic variables x_2, x_3, and x_4, as follows:

$$\begin{aligned} 2x_1 - x_5 + x_3 &= 2 \\ \frac{3x_1}{2} - \frac{x_5}{2} + x_4 &= 2 \\ -\frac{x_1}{2} + \frac{x_5}{2} + x_2 &= 3 \\ Z &= -\frac{7x_1}{2} + \frac{3x_5}{2} - 9. \end{aligned} \tag{13}$$

The basic feasible solution is $x_2=3$, $x_3=2$, $x_4=2$. (The other variables, not being in the basis, are zero.) The cost function decreases to $Z=-9$. By examining the relative cost factors of (13), it can be seen that the value of the cost function can be decreased by increasing x_1 only. The positive value of the relative cost factor of x_5 prohibits increasing x_5 from zero, since this would result in an increase for Z. Testing again to find the new basis, it can be seen that x_1 replaces x_3.

$$\begin{aligned} 2x_1 = 2; &\quad \therefore\ x_1 = 1 \\ \frac{3x_1}{2} = 2; &\quad \therefore\ x_1 = \frac{4}{3} \\ -\frac{x_1}{2} = 3; &\quad x_1 = -6. \end{aligned}$$

Notice that one value for x_1 is -6. This cannot be used, since it would result in an increase in the value

of the cost function. Eq. (13) is now put in canonical form with respect to the basic variables x_1, x_2, and x_4, as follows:

$$\begin{aligned}
\frac{x_3}{2} - \frac{x_5}{2} + x_1 &= 1 \\
-\frac{3x_3}{4} + \frac{x_5}{4} + x_4 &= \frac{1}{2} \\
\frac{x_3}{4} + \frac{x_5}{4} + x_2 &= \frac{7}{2} \\
Z &= -\frac{x_5}{4} + \frac{7x_3}{4} - \frac{25}{2}. \qquad (14)
\end{aligned}$$

The solution is $x_1=1$, $x_2=7/2$, $x_3=0$, $x_4=1/2$, and $x_5=0$; the value of Z is $-25/2$. Repeating the process,

$$\begin{aligned}
-\frac{x_5}{2} &= 1; \qquad \therefore\ x_5 = -2 \\
\frac{x_5}{4} &= \frac{1}{2}; \qquad \therefore\ x_5 = 2 \\
\frac{x_5}{4} &= \frac{7}{2}; \qquad \therefore\ x_5 = 14.
\end{aligned}$$

The new basis is x_1, x_2, and x_5, and

$$\begin{aligned}
-x_3 + 2x_4 + x_1 &= 2 \\
3x_3 + 4x_4 + x_5 &= 2 \\
x_3 - x_4 + x_2 &= 3 \\
Z &= \frac{5x_3}{2} + x_4 - 13. \qquad (15)
\end{aligned}$$

The solution is $x_3=0$, $x_4=0$, $x_5=2$, $x_1=2$, and $x_2=3$. The value taken by Z is -13. Since all of the relative cost factors are now positive, neither x_3 nor x_4 can be increased. This indicates that the above solution is one which minimizes Z. This solution agrees with the solution obtained by the graphical method. Note that the intermediate solutions are also corners of the convex polygon of Fig. 7.

Application of Simplex to Function Synthesis

The inequalities written for arbitrary functions, as described earlier, must be put into a form suitable for processing by Simplex. Since the Simplex solution yields only positive values for the unknowns, each N_j is represented as the difference of two positive numbers $(t_j - C_j)$ which would correspond to clockwise and counterclockwise windings, respectively. Hence, the solution to the function 01011101 would be of the form $t_0=2$, $C_1=2$, $t_2=1$, and $C_3=1$. Prepared for Simplex, inequalities (4) would be written as

$$\begin{aligned}
&(t_0 - C_0) && \geq 1 \\
&(t_0 - C_0) + (t_1 - C_1) && \leq 0 \\
&(t_0 - C_0) + (t_2 - C_2) && \geq 1 \\
&(t_0 - C_0) + (t_1 - C_1) + (t_2 - C_2) && \geq 1 \\
&(t_0 - C_0) + (t_3 - C_3) && \geq 1 \\
&(t_0 - C_0) + (t_1 - C_1) + (t_3 - C_3) && \leq 0 \\
&(t_0 - C_0) + (t_2 - C_2) + (t_3 - C_3) && \geq 1 \\
&(t_0 - C_0) + (t_1 - C_1) + (t_2 - C_2) + (t_3 - C_3) && \leq 0, \qquad (16)
\end{aligned}$$

where t_j and C_j are the unknowns. The I_n coefficients have the value of unity and are therefore dropped.

Note that in addition to the $\leq$ signs appearing in inequalities (16), the $\geq$ signs also appear. In the case of $\geq$ constraints, a slack variable is subtracted and a dummy variable is added to the left-hand side of the inequality. Dummy variables are required to provide the initial basis. The initial basis is then comprised of slack and dummy variables, namely d_0, S_1, d_2, d_3, d_4, S_5, d_6, and S_7, as shown in (17).

$$\begin{aligned}
&(t_0-C_0) && -S_0+d_0=1 \\
&(t_0-C_0)+(t_1-C_1) && +S_1 \quad =0 \\
&(t_0-C_0)+(t_2-C_2) && -S_2+d_2=1 \\
&(t_0-C_0)+(t_1-C_1)+(t_2-C_2) && -S_3+d_3=1 \\
&(t_0-C_0)+(t_3-C_3) && -S_4+d_4=1 \\
&(t_0-C_0)+(t_1-C_1)+(t_3-C_3) && +S_5 \quad =0 \\
&(t_0-C_0)+(t_2-C_2)+(t_3-C_3) && -S_6+d_6=1 \\
&(t_0-C_0)+(t_1-C_1)+(t_2-C_2)+(t_3-C_3) && +S_7 \quad =0. \qquad (17)
\end{aligned}$$

The cost function is

$$\begin{aligned}
Z = {} & t_0 + t_1 + t_2 + t_3 + C_0 + C_1 + C_2 + C_3 \\
& + Md_0 + Md_2 + Md_3 + Md_4 + Md_6. \qquad (18)
\end{aligned}$$

Note that dummy variables are included in the cost function. The coefficients M of the dummy variables are chosen very large to bias the problem so that, if possible, the dummy variables will not be included in the basis which minimizes Z.[8]

Eq. (18) is rearranged in canonical form by appropriate substitutions from (17), eliminating the dummy variables, such that

$$\begin{aligned}
Z = {} & (1-5M)t_0 + (1-M)t_1 + (1-3M)t_2 \\
& + (1-2M)t_3 + (1+5M)C_0 + (1+M)C_1 \\
& + (1+3M)C_2 + (1+2M)C_3 + MS_0 \\
& + MS_2 + MS_3 + MS_4 + MS_6 + 5M. \qquad (19)
\end{aligned}$$

The problem is now ready to be treated in the manner previously described.

[8] In some Simplex applications, the existence of a dummy variable in the final basis indicates that there is no feasible solution. In our application, this type of result simply indicates that more than one core is required to mechanize the given function.

Residue Functions

It will be recalled that if a function h were processed by Simplex and dummy variables appeared in the solution, then a multicore configuration is required to mechanize the function. Further, if one core were wound in accordance with the resulting values of t_j and C_j, part of the function would be obtained, and a new function would be defined by pacing a ONE in the positions indicated by the subscripts of the dummy variables in the solution. The portion of the function initially mechanized will be referred to as f, and the remaining portion as the residue function g. Eq. (17) may be written as follows:

$$\text{when } h_i = 1, \quad \sum_j (t_j - C_j) - S_i + d_i = 1,$$

$$h_i = 0, \quad \sum_j (t_j - C_j) + S_i = 0.$$

Let $F_i = \sum_j (t_j - C_j)$; hence, when $F_i \geq 1$, $f_i = 1$, and when $F_i \leq 0$, $f_i = 0$.

The symbol [ ] is defined as

$$[X] = 1, \quad \text{if } X \geq 1$$

$$[X] = 0, \quad \text{if } X \leq 0.$$

The residue function is defined as

$$[G_i] = 1, \quad \text{when } d_i > 0 \tag{20}$$

$$[G_i] = 0, \quad \text{when } d_i = 0, \quad \text{or when there is no } d_i. \tag{21}$$

Eq. (8) may now be written as

$$h_i = [F_i] + [G_i]. \tag{22}$$

To prove (22), all possible combinations will be examined; *i.e.*,

Case 1: $h_i = 1$, $[F_i] = 1$
Case 2: $h_i = 1$, $[F_i] = 0$
Case 3: $h_i = 0$, $[F_i] = 1$
Case 4: $h_i = 0$, $[F_i] = 0$.

Case 1

In this case, $h_i = 1$; $[F_i] = 1$. By definition,

$$F_i \geq 1. \tag{23}$$

Since Simplex guarantees that the original set of equations must be satisfied and that all variables are positive, we may write

$$F_i - S_i + d_i = 1$$

$$S_i \geq 0. \tag{24}$$

Adding (23) and (24),

$$2F_i + d_i \geq 2 \quad \text{or} \quad F_i \geq \frac{2 - d_i}{2}. \tag{25}$$

Hence, where $h_i = 1$ and $F_i \geq 1$, the dummy variable must take the value of zero (since all variables are positive) and we may state, from (21) that $[G_i] = 0$.

Case 2

In this case, $h_i = 1$; $[F_i] = 0$. By definition

$$F_i \leq 0, \tag{26}$$

and we may write

$$F_i - S_i + d_i = 1$$

$$-F_i \geq 0$$

$$S_i \geq 0. \tag{27}$$

Adding inequalities (27),

$$d_i \geq 1 \quad \text{or} \quad d_i > 0. \tag{28}$$

Hence, where $h_i = 1$ and $[F_i] = 0$, dummy variables must remain and we may state, from (20), that $[G_i] = 1$.

Case 3

In this case $h_i = 0$; $[F_i] = 1$. By definition,

$$F_i \geq 1 \tag{29}$$

and

$$F_i + S_i = 0. \tag{30}$$

In this case, (30) could be satisfied only if S_i were negative, which is not permitted by Simplex. Hence, Case 3 is not possible.

Case 4

In this case, $h_i = 0$; $[F_i] = 0$. By definition,

$$F_i \leq 0. \tag{31}$$

In this case, (30) is satisfied by a positive slack variable which, of course, is permissible. Since there is no d_i, however, we may state, from (21), that $[G_i] = 0$. Thus (22) is satisfied under all possibile combinations.

The residue function defined above is now treated by Simplex. If it is a one-core function, the problem is terminated yielding a two-core function. If in processing the residue function dummy variables remain in that solution, then a new residue function is defined.

The above procedure is repeated until the final residue function processed contains no dummy variables. Therefore, one magnetic core is required for each function or residue function processed in the course of a problem. From another point of view, it may be said that the given function in a problem is reduced to a set of one-core functions, each of which might be represented by a consistent set of inequalities. To obtain the original function, all of these one-core functions are ORed together by the output line.

Simplex Modifications

In the course of the applications of the Simplex algorithm to the above problem, several difficulties arose which resulted in some modifications. For example, some solutions yielded the minimum total number of turns at the expense of additional cores. An ex-

treme example of this type of difficulty is the result calling for zero turns and a residue function equal to the original function. The type of Simplex algorithm finally developed is "dummy-oriented"; *i.e.*, in every applicable decision process during the course of the problem, dummy variables are removed from the basis. The following modifications were introduced:

1) Once a dummy leaves the basis, it is not permitted to return. This is accomplished by ignoring the relative cost factors of dummy variables in the cost function.

2) The problem is not necessarily permitted to end when all of the relative cost factors become positive. Instead, the initial conditions of the problem are altered by changing the value of a relative cost factor assigned to one of the dummy variables so that, at this stage of the problem, one of the nondummy variables assumes a negative relative cost factor. The test described previously is applied to determine which variable, if any, will leave the basis. It is possible that a variable with a negative relative cost factor cannot enter the basis, since the test for minimum positive value yields only negative numbers. It should be noted that it is not difficult to change the initial conditions in the manner described above, because only the cost equation is affected throughout the problem.

This procedure may be explained from the geometrical point of view. Consider the cost function of (10). If the cost function were changed to $Z=-2x_1-30x_2$, then a different corner of the convex polygon would minimize the new cost function.

Summarizing, it may be stated that the relative cost factors are repeatedly altered, forcing the problem to continue until a basis containing a minimum number of dummy variables is obtained. At the price of a more "expensive" solution (*i.e.*, one which might contain more turns but uses fewer cores), the problem is permitted to terminate only when it is impossible to remove dummy variables from the basis.

It is interesting to note that it is always possible to remove at least one dummy from the initial basis. The partial result would be a function containing a single ONE and a residue function with one less ONE than the original function. All Boolean functions containing a single ONE are realizable with a single core.

Results

The Simplex algorithm described in this paper for synthesizing Boolean functions was programed for the Burroughs 220 computer. Many randomly selected functions of six or fewer variables were successfully synthesized. Several nets of cores were wound in accordance with the Simplex-derived results, and were operated successfully in the laboratory.

Future Work

Since the writing of this paper, an improved Simplex algorithm has been developed.[9] Problems involving nine variables have been successfully processed. In addition, provisions for handling "don't care" conditions have been added. This work will be reported in a future paper.

There is one important aspect that must be given consideration. Initially, and to avoid complicating the problem, no upper limit has been placed on the total number of input turns. Let M ampere-turns be in the set direction, and N ampere-turns in the reset direction, and let us depend upon a condition of $M-N=1$ to set the core. Clearly, as the values of M and N increase, the worst-case tolerance requirements on the current drivers become more severe. It will be recalled that finding the minimizing basic feasible solution is an iterative procedure in which many basic feasible solutions are considered. Each of these solutions, of course, satisfies the original system of equations. This suggests the possibility of storing these solutions, and of considering each one in the light of the above-mentioned current-driver tolerance. Thus, a way may be provided for relieving the tolerance requirement by a sacrifice in the total core count.

Acknowledgment

The author appreciates the contributions of J. Celia of the Burroughs Corporation, who programed the problem for the Burroughs 220, and also the assistance of N. Papantonis.

[9] Based on suggestions by R. Gomory, IBM Corp., a panelist at the 15th Natl. Conf. of the Assoc. for Computing Machinery, Milwaukee, Wis.; 1960.

An Algorithm for Automatic Design of Logical Cryogenic Circuits*

E. H. Sussenguth, Jr.†, member, ire

Summary—Logical cryogenic circuits differ from the circuitry used with other devices as paths for both the required function and its denial must be provided. Present techniques for logical design of cryogenic circuits either rely on the experience of the designer to achieve a minimal circuit or are derived from analogous relay circuits. An algorithm to develop minimal (or near minimal) circuits by collapsing a complete decoding-tree structure by removing unneeded devices is discussed. The algorithm requires no subjective decisions and is readily programed for inclusion in an automatic design system.

A generalization of the algorithm to include functions of *n*-ary input variables and multiple outputs is also presented.

I. Introduction

In computing machines built using relay, vacuum-tube, transistor, resistor-diode, or magnetic-core logical circuitry, the component cost was such that the saving of only one or two devices per circuit could substantially reduce the over-all system cost. The time and effort involved in using experienced logical designers to insure that each circuit was truly minimal was repaid with a reduced expense for components. The cryotron, however, is no more expensive than the wire used to connect it and is manufactured by evaporation techniques in which an entire processing unit, rather than an individual component, may be constructed at one time. For the cryotron and similar devices an automated logical design system may be efficiently used and the system need not require an absolutely minimal circuit. The algorithm described below sacrifices truly minimum results for a method readily adaptable to automatic computation.

II. Physical and Logical Restrictions of Cryogenic Circuits

The current in an electric circuit comprising a current source and two parallel resistive paths will divide in a ratio inversely proportional to the resistances. If one of these resistances is set to zero, the resistance ratio becomes infinite and all of the current is diverted to the path which contains the resistanceless (superconducting) element. If, subsequently, the second resistance is set to zero, the current will persist in the first path as there is no energy available to induce a switching (or dividing) action. A resistance must be inserted into the first path before the current will switch to the second path. A circuit of this type remembers which element was made resistanceless first by the presence or absence of current and is a useful digital computer component.

The cryotron, a device which may be rapidly switched between resistive and superconducting states, utilizes these principles in digital circuitry. Logically the individual cryotron realizes $A\overline{B}\rightarrow C$ where A is the statement, "A current source is available to the cryotron gate," and B is the statement, "There is current (of sufficient magnitude) in the cryotron control," and C is "There is current in the path of which the cryotron gate is a part." There is, however, an important restriction to this logical representation: a superconducting parallel connection is required to divert the current from the gate wire when current is applied to the control wire. If a zero-resistance by-pass is not provided, the gate current will not be inhibited but will merely divide according to the resistance ratio and form I-R voltage drops.

It will be noted that the normally closed relay realizes the same logical statement $A\overline{B}\rightarrow C$, where A now represents, "A current (or voltage) source is available to the relay contacts," B, "There is power (of sufficient magnitude) applied to the relay winding," and C, "There is current in the path of which the relay contacts are a part." However, the restriction of a parallel connection does not apply to relay circuits: when power is applied to the relay winding (corresponding to the cryotron control), an open circuit results in the contact (gate) circuitry and signal flow (current) is absolutely inhibited.

In the design of logical cryogenic circuits, therefore, it is not sufficient to build only a circuit which realizes the desired function f; rather, to achieve the desired switching action, paths for both f and $\bar{f}$ must be provided.

A second restriction, which may or may not be important according to the particular application, is that if the by-pass and the cryotron are both superconducting, a current, subsequently applied, will divide between the paths in a ratio inversely proportional to the path inductances. If either of these paths is used to control other cryotrons, the divided current may not have sufficient magnitude to insure switching. If, however, the paths rejoin before any control action is attempted, parallel superconducting connections may be employed. This restriction may be overcome by designing circuits in which there is one and only one superconducting path

* Received by the PGEC, November 21, 1960; revised manuscript received, March 22, 1961. This work was supported in part by the Bureau of Ships, Department of the Navy.

† IBM Corp. Res. Lab., Yorktown Heights, N. Y.

joining the input terminal and an output terminal. Such circuits will be called *disjunctive* circuits.

Cryogenic circuits are manufactured by evaporating metals through a masking plate onto a plane and it is desirable to use as few masks as possible. The number of masks required to construct a circuit depends on the topological complexity of the circuit for every crossing must be insulated by the evaporation of a dielectric. This, then, may be a third restriction on the design of cryogenic circuits, albeit a practical rather than logical one. The problem of minimizing the crossings in a circuit is very difficult but may be of equal importance with the minimization of the device count. One possible compromise is to design so that the control lines corresponding to different variables do not cross, that is, all of the cryotrons controlled by one variable will be in a single *column*.

An advantage of cryogenic circuitry is that the loading problems familiar to many other technologies do not occur; any number of cryotrons may be controlled by the current from one source for there is no power loss in the steady state as current flows only in superconducting paths. The losses which occur in the transient state[1] do not seem to be significant enough to impose limitations on the circuit design (but may, however, be a major factor in determining the packing density of cryogenic circuits).

These are the essential differences between cryogenic circuits and circuits made with other current-switching devices and indicate a possible need for a new synthesis technique. Two different design forms are apparent (Fig. 1): in the first, two distinct circuits C_1, realizing f, and C_2, realizing $\bar{f}$, are constructed; in the second, one circuit C_3 which simultaneously realizes both f and $\bar{f}$ is needed. The circuits C_1 and C_2 may be designed using techniques similar to those developed for normally closed relays. Most of these methods depend on a few primary rules to determine a basic structure and then rely on the experience and insight of the designer to achieve a minimum (*i.e.*, minimum device count) circuit.

The circuit C_3 is essentially a disjunctive tree modified for two outputs which may be collapsed by removing unnecessary cryotrons. These collapsed tree circuits may be designed with an algorithm which may be readily adapted to automatic computation. We have found that in a great majority of the cases tested fewer cryotrons are required for the collapsed tree realization than for the system requiring separate f and $\bar{f}$ circuits. In the collapsed tree circuits there is one and only one superconducting path for a particular state of the input variables.

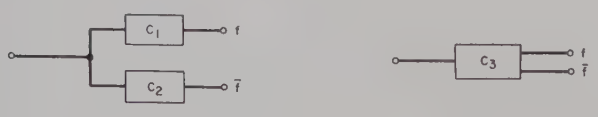


Fig. 1—Two possible circuit forms for realizing f and $\bar{f}$.

The design algorithm will be described first for the specific case of cryogenic circuits and then, more briefly, in a general form. First, however, a notation is presented which simplifies the application of the algorithm if the rules are carried out by hand and indicates a method for storing the information when the algorithm is programmed for a digital computer.

III. Notation

The basic cryotron pair used in the disjunctive tree is shown in detail in Fig. 2(a) and in block diagram form in Fig. 2(b). For simplicity we shall understand that the upper terminal always represents the condition $X=0$ and the lower terminal $X=1$. To distinguish different X boxes, a subscript will be appended to the control variable designator. Fig. 3(a) shows a complete circuit in block diagram form. Fig. 3(b) indicates a possible method of compactly denoting the same structure: the heading of each column indicates the control variable for the column, the ith line in the column represents block number i, the first entry in the ith line indicates to which block (or output) the upper terminal is connected, and the second entry indicates to which block the lower terminal is connected.

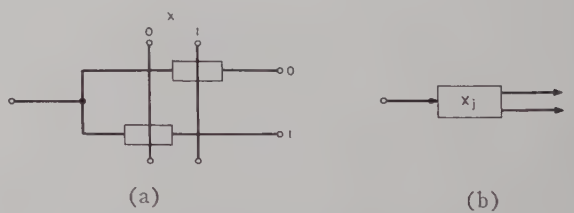


Fig. 2—The basic cryotron pair structure (a) may be simplified to the block diagram form (b).

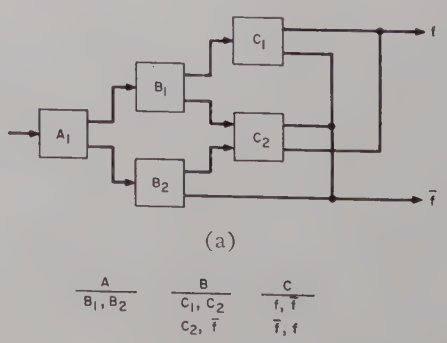


Fig. 3—A circuit realizing f and $\bar{f}$ where $f=\bar{A}\bar{B}\bar{C}+ABC+A\bar{B}C$. (a) In block diagram form. (b) In compact notation.

[1] M. K. Haynes, "Transient analysis of cryotron networks by computer simulation," Proc. IRE, vol. 49, pp. 245–257; January, 1961.

IV. The Algorithm for Single Output Cryogenic Circuits

The method of design assumes a disjunctive or decoding tree structure[2] and collapses the tree by removing unnecessary cryotron pairs. The reduction is accomplished by implicit use of the identities

$$XY + \overline{X}Y = Y$$

and

$$XZ + YZ = (X + Y)Z.$$

A. Problem Statement

Given a finite number, n, of binary input variables $A_1, A_2, \cdots, A_n$, realize the Boolean function f and its complement $\bar{f}$ where $f=f(A_1, A_2, \cdots, A_n)$. The synthesized circuit is to use as few cryotron-type devices as possible. There shall be one and only one path through the circuit for a particular state of the input variables.

B. Minimization Rules

Form a complete disjunctive tree in the n variables. (For descriptive purposes assume the tree opens to the right.) The 2^n terminals of the nth level are connected to either the f output terminal or the $\bar{f}$ output as required by the function f. Apply the following rules for each variable in turn: $A_n, A_{n-1}, \cdots, A_2, A_1$. The process may be terminated if upon reaching some variable no modification can be made.

1) *Rule 1:* Examine the connections of the individual logical units of the ith variable separately. If both output branches are connected to the same element on the right, the logical unit may be removed and its input branch connected to the element to which the output branches were previously joined. If the output branches are joined to different elements on the right, no modification is made.
2) *Rule 2:* Examine the connections of the logic units of the ith variable in pairs taken in all possible ways. If the units are identically connected, that is if both upper branches are connected to the same element on the right and both lower branches are connected to the same element (different from the upper) on the right, one logic unit may be removed and its input branch connected to the input branch of the second logic unit. If the pairs are not identically connected, no modification is made.

These rules develop the circuit with a minimum device count for the particular permutation of input variables specified. (A proof of minimality is given in the Appendix.) Since different permutations result in different circuits, it is necessary to test all permutations of the input variables to insure that the minimum-device-count circuit for the required function is realized.

An example in four variables showing the application of the rules is given in Fig. 4. Both the block diagram and the compact notation are shown for each step. Fig. 5 indicates a circuit realizing the same function but using a different permutation of the input variables.

The necessity of testing all permutations of the input variables is a limitation of the procedure, and even with a high speed computer this may be quite time consuming if the function is in many variables. We have been unable to discover a technique which will indicate the best permutation by examination of the structure of the required function. However, a preliminary examination of the function may be made to discover any symmetries and reduce the effective number of permutations. We have also found that in general several permutations of the input variables lead to minimal circuits.[3] This fact may enable one to use a directed search[4] for problems of many variables if the designer wishes to save computing time and is willing to accept a result which is not necessarily minimum. Briefly, the directed search method is this: after an arbitrary initial order and circuit, two variables, selected randomly, are interchanged; if the new circuit is an improvement, the order is retained and another interchange tried; if there is no improvement, the previous order is restored and another interchange tried. The process is terminated when a specified threshold has been achieved or when a specified number of consecutive failures occurs.

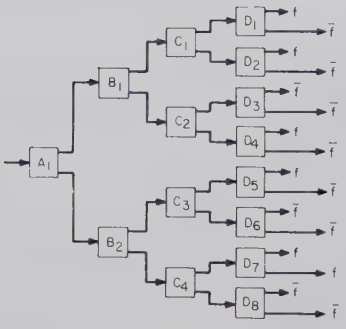
The algorithm described above (not including the directed search) has been programmed for the IBM 704. Table I indicates the running time for circuits of various numbers of variables. The time includes algorithm application for all permutations. It is apparent from the table that for circuits of seven or eight variables a technique other than exhaustion of the permutations would be valuable, even though this technique is itself moderately complex.

The collapsed tree method defines a circuit which is minimum only within the strict form of a disjunctive tree. Modifications of the tree structure which allow control leads to be interwoven may develop circuits with fewer devices (Fig. 6), and, if the restriction regarding disjunctive paths is removed, special constructions may

[2] S. H. Washburn, "Relay trees and symmetric circuits," *Trans. AIEE*, vol. 68, (*Commun. and Electronics*, pt. I), pp. 582–586; 1949.

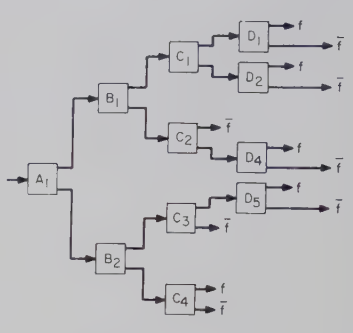
[3] The 2^{2^4} functions of 4 variables may be reduced to 402 representative function classes when equivalences under permuting and negating the input variables are removed; for each circuit there are, of course, $4!=24$ possible permutations of the input variables. These 402 distinct functions of 4 variables have been tested with the IBM 704 program; only 18 were found to have unique minimal circuits and the average number of permutations which resulted in a minimal circuit was 7.6.

[4] B. Dunham, *et al.*, "The multipurpose bias device, part II, the efficiency of logical elements," *IBM J. Res. & Dev.*, vol. 3, pp. 46–53; January, 1959.



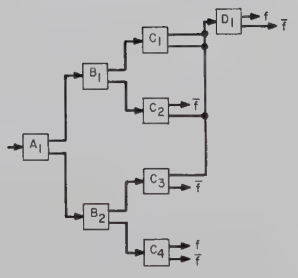
(a)

A	B	C	D
B_1, B_2	C_1, C_2	D_1, D_2	$f, \bar{f}$
	C_3, C_4	D_3, D_4	$f, \bar{f}$
		D_5, D_6	$f, \bar{f}$
		D_7, D_8	$f, \bar{f}$
			$f, \bar{f}$
			$f, \bar{f}$
			$f, \bar{f}$
			$f, \bar{f}$



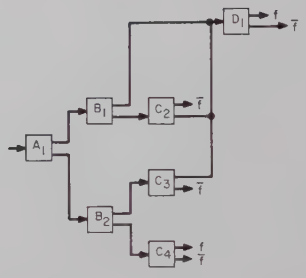
(b)

A	B	C	D
B_1, B_2	C_1, C_2	D_1, D_2	$f, \bar{f}$
	C_3, C_4	f, D_4	$f, \bar{f}$
		$D_5, \bar{f}$	—
		$f, \bar{f}$	$f, \bar{f}$
			$f, \bar{f}$
			—
			—
			—



(c)

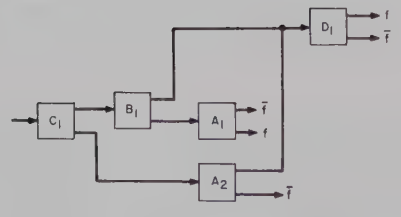
A	B	C	D
B_1, B_2	C_1, C_2	D_1, D_1	$f, \bar{f}$
	C_3, C_4	$\bar{f}, D_1$	—
		$D_1, \bar{f}$	—
		$f, \bar{f}$	—
			—
			—
			—
			—



(d)

A	B	C	D
B_1, B_2	D_1, C_2	—	$f, \bar{f}$
	C_3, C_4	$\bar{f}, D_1$	—
		$D_1, \bar{f}$	—
		$f, \bar{f}$	—
			—
			—
			—
			—

Fig. 4—Steps in the minimization of the function $f=AB\bar{C}+\bar{A}C\bar{D}+\bar{B}\bar{C}\bar{D}$. (a) The complete tree. (b) After rule 1 is applied to the D column. (c) After Rule 2 is applied to the D column. (d) After Rule 1 is applied to the C column.



C	B	A	D
B_1, A_2	D_1, A_1	$\bar{f}, f$	$f, \bar{f}$
		$D, \bar{f}$	

Fig. 5—The permutation (C, B, A, D) yields a circuit with fewer blocks.

TABLE I

COMPUTATION TIME t FOR A CIRCUIT OF n VARIABLES, $t \approx (2^{n+1}+50)n!$ msec

n	t
2	0.1 sec
3	0.4 sec
4	2.0 sec
5	13.0 sec
6	2.0 min
7	26.0 min
8	6.0 hr

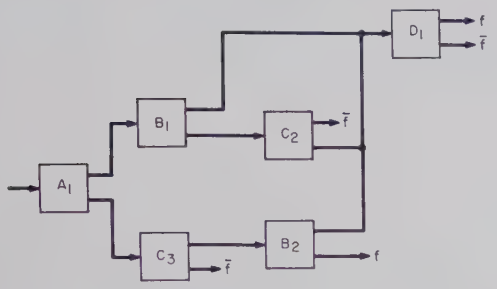


Fig. 6—The modified tree which is the result of applying Rule 3 to the C column of the tree of Fig. 4(c).

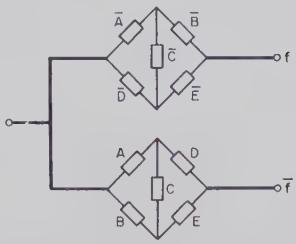


Fig. 7—A bridge-type circuit realizes the function $f=AB+ACE+BCD+D\bar{E}$ with six fewer cryotrons than the minimal circuit designed by the algorithm but the bridge circuit violates the disjunctive-path requirement.

reduce the device count significantly (Fig. 7). Modifications of the former type may be determined mechanically if a third rule is added to the two given above.

3) *Rule 3:* For all the logic units of the ith variable examine in pairs those which are connected from the same unit of the $(i-1)$th variable on the left. For each pair so connected examine the output connections. If both upper branches (or both lower branches) are connected to the same unit on the right, the tree is modified by interchanging the unit of the $(i-1)$th variable with one of the units of the ith and removing the second unit of the ith variable. The upper (lower) terminal of the ith variable logic unit (which is now in the $(i-1)$th level) is connected to the element which was previously common; the lower (upper) terminal is connected to the $(i-1)$th unit which has just been transferred to the ith level. The upper terminal of the $(i-1)$th unit (now in the ith level) is connected to the noncommon element of the old upper unit and the lower terminal to the noncommon element of the old lower unit. If there is no pair of units identically connected on the left or if such pairs exist but do not have upper or lower output connections in common, no modification is made.

Rule 3 is applied after a circuit has been designed using Rules 1 and 2. An illustration is given in Fig. 6. We have found that reductions of this type are rather infrequent once a minimal circuit has been constructed using Rules 1 and 2 and all permutations tested.[5]

V. General Form of the Algorithm

The minimization procedure described above may be stated in more general terms: input and output variables are not limited to binary states but many have a finite number of allowed states.

A. Problem Statement

Given a finite number n of input variables $A_1, A_2, \cdots, A_n$, each input variable may assume one of a finite number k_i of stable states:

$$A_1\colon a_{11}, a_{12}, \cdots, a_{1,k_1}$$

$$A_2\colon a_{21}, a_{22}, \cdots, a_{2,k_2}$$

$$\cdots\cdots\cdots\cdots$$

$$A_n\colon a_{n1}, a_{n2}, \cdots, a_{n,k_n}.$$

With each A_i, we associate a logical block with one input branch and k_i output branches; the block is such that output branch j is active if and only if the input branch is active and A_i is in state a_{ij}. The output function F has a finite number, l, of stable states: $f_1, f_2, \cdots, f_l$, where each

$$f_i = f_i(A_1, A_2, \cdots, A_n).$$

We are to construct a circuit which realizes F using as few logical blocks as possible.

B. Minimization Rules

Form a disjunctive tree in the n variables. Mark (*i.e.*, connect) the terminals of the nth variable units with symbols representing the appropriate state of F.

Apply the following three rules for each variable in turn: $A_n, A_{n-1}, \cdots, A_2, A_1$. For the ith variable apply the rules repetitively (*i.e.*, 1, 2, 3, 1, 2, 3, $\cdots$) until no further modifications can be made; then proceed to variable $i-1$. If upon reaching some variable no modification can be made, the entire process may be terminated.

1) *Rule 1:* Examine each unit of the ith variable separately. If all output terminals are identically marked, remove the unit and change the input terminal marking to the symbol which was associated with the output terminal. If the symbols are not identical, no modification is made.

2) *Rule 2:* Examine the units of the ith variable in pairs. If the sets of output terminals are identical remove one unit and change its input terminal marking to the symbol representing the second unit. If the sets differ, no modification is made.

3) *Rule 3:* Examine each unit of the ith variable separately. If all the output terminals contain a common symbol, remove that symbol from the out-

[5] Only 6 of the 402 minimal circuits for 4-variable functions could be reduced by application of Rule 3.

put terminals of the unit and append the symbol to the input terminal branch. If there is no common symbol, no modification is made.

Notice that Rule 3 is a modification of Rule 1: for Rule 1 the output terminal symbols are treated as an unordered set, for Rule 3 the output terminal symbols are treated individually, that is; if $f_1f_2f_3$ is a particular terminal marking, for Rule 1 $f_1f_2f_3$ is considered as one symbol, for Rule 3 f_1, f_2, and f_3 are considered individually. If all of the f_i are independent (*i.e.*, $f_i \cap f_j = 0$ for all $i \neq j$), Rule 3 will never be needed and may be omitted.

The example of Fig. 8 is carried through step by step using the compact notation; the initial and final circuits are also shown in block diagram form. The compact notation and block diagram structure have been extended in the obvious manner.

VI. Summary

An algorithm is presented which is useful in the automatic synthesis of logical circuits utilizing cryotron-type components. In its most general form the algorithm applies three easily programmed rules to remove unneeded components from a disjunctive tree circuit to form a collapsed tree structure with a minimal device count. Although particular constructions may develop circuits with a smaller device count, if the device is inexpensive, a hand design may be ultimately more costly.

Inputs:

A: a_1, a_2, a_3, a_4
B: b_1, b_2
C: c_1, c_2, c_3

Output required:

F: f_1, f_2, f_3 (0 denotes no connection)

$$f_1 = a_1b_1c_1 + a_1b_1c_2 + a_2b_1c_2 + a_2b_2c_2 + a_3b_1c_1 + a_3b_1c_2 + a_4b_1c_1 + a_4b_1c_2 + a_4b_1c_3 + a_4b_2c_1 + a_4b_2c_2 + a_4b_2c_3$$

$$f_2 = a_1b_1c_2 + a_1b_2c_1 + a_1b_2c_2 + a_1b_2c_3 + a_2b_1c_3 + a_2b_2c_3 + a_3b_1c_2 + a_3b_2c_1 + a_3b_2c_2 + a_3b_2c_3$$

$$f_3 = a_2b_1c_1 + a_2b_1c_2 + a_2b_1c_3 + a_4b_1c_1 + a_4b_1c_2 + a_4b_1c_3$$

(a) The complete tree

A	B	C
B_1, B_2, B_3, B_4	C_1, C_2	$f_1, f_1f_2, 0$
	C_3, C_4	f_2, f_2, f_2
	C_5, C_6	f_3, f_1f_3, f_2f_3
	C_7, C_8	$0, f_1, f_2$
		$f_1, f_1f_2, 0$
		f_2, f_2, f_2
		f_1f_3, f_1f_3, f_1f_3
		f_1, f_1, f_1

(b) Rule 1 applied to the C variable

A	B	C
B_1, B_2, B_3, B_4	C_1, f_2	$f_1, f_1f_2, 0$
	C_3, C_4	—
	C_5, f_2	f_3, f_1f_3, f_2f_3
	f_1f_3, f_1	$0, f_1, f_2$
		$f_1, f_1f_2, 0$
		—
		—
		—

(c) Rule 2 applied to the C variable

A	B	C
B_1, B_2, B_3, B_4	C_1, f_2	$f_1, f_1f_2, 0$
	C_3, C_4	—
	C_1, f_2	f_3, f_1f_3, f_2f_3
	f_1f_3, f_1	$0, f_1, f_2$
		—
		—
		—
		—

(d) Rule 3 applied to the C variable

A	B	C
B_1, B_2, B_3, B_4	C_1, f_2	$f_1, f_1f_2, 0$
	C_3f_3, C_4	—
	C_1, f_2	$0, f_1, f_2$
	f_1f_3, f_1	$0, f_1, f_2$
		—
		—
		—
		—

(e) Rule 2 applied to the C variable

A	B	C
B_1, B_2, B_3, B_4	C_1, f_2	$f_1, f_1f_2, 0$
	C_3f_3, C_3	—
	C_1, f_2	$0, f_1, f_2$
	f_1f_3, f_1	—
		—
		—
		—
		—

(f) Rule 2 applied to the B variable

A	B	C
B_1, B_2, B_1, B_4	C_1, f_2	$f_1, f_1f_2, 0$
	C_3f_3, C_3	—
	—	$0, f_1, f_2$
	f_1f_3, f_1	—
		—
		—
		—
		—

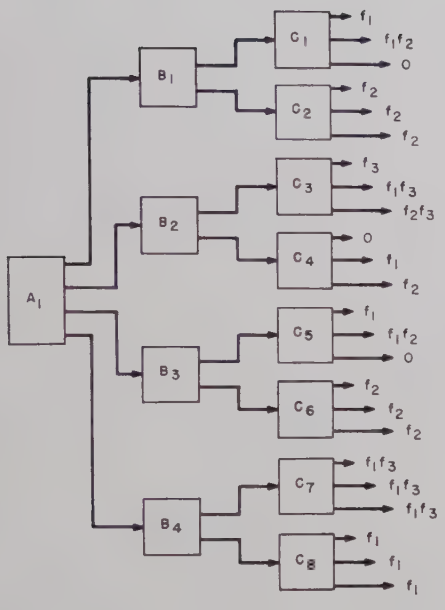
(g) Rule 3 applied to the B variable

A	B	C
B_1, B_2C_3, B_1, B_4f_1	C_1, f_2	$f_1, f_1f_2, 0$
	$f_3, 0$	—
	—	$0, f_1, f_2$
	$f_3, 0$	—
		—
		—
		—
		—

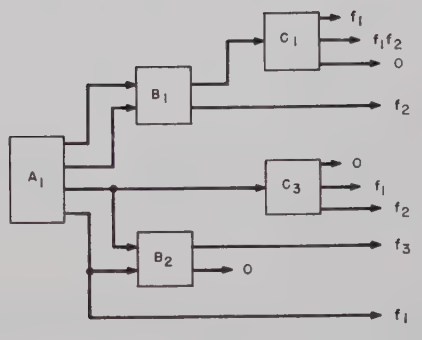
(h) Rule 2 applied to the B variable

Fig. 8. cont.

A	B	C
B_1, B_2C_3, B_1, B_2f_1	C_1, f_2	$f_1, f_1f_2, 0$
	$f_3, 0$	$0, f_1, f_2$



(i) The complete tree [corresponds to (2)].



(j) The collapsed tree [corresponds to (h)].

Fig. 8—Problem statement.

Appendix

Theorem

The single output cryogenic circuits designed by Rules 1 and 2 (Section IV-B) of the algorithm are disjunctive and have a minimum device count for that class of circuits in which the variable control lines are not interwoven.

Proof of disjunctiveness: The tree from which the algorithm is begun is disjunctive. To destroy the disjunctiveness it is necessary to form a branch opening to the right, both parts of which may be conducting at the same time. Rule 1 removes one switching unit but forms no inherently new connections and, hence, no branches (Fig. 9), next page. Rule 2 removes one unit and makes a new connection, but this new connection does not form a branch opening to the right (Fig. 9). As no other modifications are made, no branches are formed, and the disjunctiveness is preserved.

Proof of minimality: Consider two circuits, I and II, which realize the same Boolean function f and its denial $\bar{f}$. Circuit I is designed by using Rules 1 and 2 of the algorithm. Circuit II is a disjunctive circuit with variables occurring in the same order as I and having the same columnar form of control. Assume Circuit II has fewer logic units than Circuit I and suppose the order of the variables is $P, Q, \cdots, Z$.

Either circuit may be divided into two parts by cutting all the input lines to the logic units of some variable, say V. That part of the circuit to the left of the cut must develop the 2^k terms in the k variables $P, Q, \cdots, U$ and, as the circuits are disjunctive, each term is carried on one and only one line (but, of course, each line may carry several terms). Each term T may or may not be operated on by the variable V according as the line on which it is carried is an input for a switching unit of variable V (a V unit) or is not an input to a V unit. After passing through the V level of logic, the terms, which now may be represented as TV and $T\overline{V}$, are operated on by the remaining variables $W, X, \cdots, Z$ in the part of the circuit to the right of the V column. If the two circuits are to realize the same function, the logical action on the terms TV and $T\overline{V}$ caused by the switching units of the variables $W, X, \cdots, Z$ must be the same in both circuits, although, of course, the detailed structure of the left parts of the circuits may differ.

As Circuit II has fewer logic units than Circuit I, for at least one variable, say V, Circuit II has fewer V units. Now cut both circuits at the input to the V level and consider the 2^k terms in the variables $P, Q, \cdots, U$. They may be partitioned into four distinct classes:

Type a: The term is carried on a line which is not connected to a V unit in either Circuit I or II.

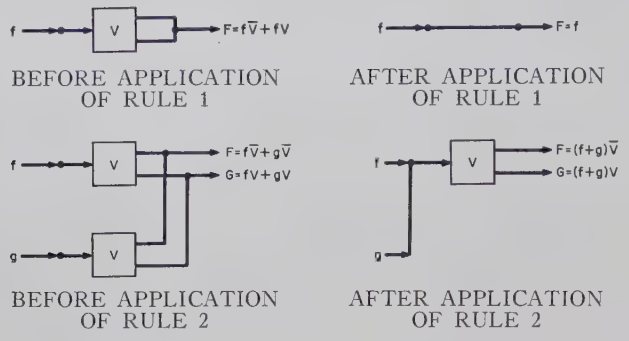


Fig. 9—Schematic representation of Rules 1 and 2.

Type b: The term is carried on a line which is not connected to a V unit in Circuit I but is connected to a V unit in Circuit II.

Type c: The term is carried on a line which is connected to a V unit in Circuit I but is not connected to a V unit in Circuit II.

Type d: The term is carried on a line which is connected to a V unit in both Circuits I and II.

By considering these classes of terms, we shall show that Circuit II cannot have fewer switching units than Circuit I.

Terms of Type *a* are those which are not influenced by the variable V and, as they play no role in the unit count, are of no interest to us.

Terms of Type *b* require that Circuit II have at least as many V units as Circuit I (not necessarily more for one V unit may switch several terms) and, hence, tend to make Circuit I have fewer logical units than Circuit II. As this contradicts the hypothesis, we do not need to consider this type.

Terms of Type *c* may require more units in Circuit I than II. However, Type *c* means that variable V does not operate on term T in Circuit II. But as Circuits I and II realize the same function, term T must be switched in a similar way in Circuit I. That is, in Circuit I, TV and $T\overline{V}$ must be identically connected to the part of the circuit to the right of variable V. But is is exactly this redundancy that is removed by Rule 1, and, hence, there cannot be any terms of this type in Circuit I. Thus there are no terms of Type *c*.

Terms of Type *d*, then, must account for the difference in the V unit count. That is, there must be at least one V unit in Circuit II which services more terms than a counterpart in Circuit I. Assume terms $T_1, T_2, \cdots, T_k$ are switched by one unit of Circuit II; its switching action may be represented by $(T_1+T_2+\cdots+T_k)V=F$ and $(T_1+T_2+\cdots+T_k)\overline{V}=G$. In Circuit I each T_i $(i=1, 2, \cdots, k)$ must be switched by some V unit (because T_i is Type *d*) and the action may be represented by T_iV and $T_i\overline{V}$. But the part of Circuit I to the right of the V column must logically switch each T_iV and $T_i\overline{V}$ in the same way that Circuit II does. Circuit II accomplishes this by forming F and G. Hence it must be possible to form F and G in Circuit I by $T_1V+T_2V+\cdots+T_kV$ and $T_1\overline{V}+T_2\overline{V}+\cdots+T_k\overline{V}$. But this type redundancy is removed by Rule 2 and, therefore, it is not possible to find units in II which service more terms than a corresponding unit of 1. Finally, then, there cannot be fewer V units in Circuit II than in Circuit I with respect to Type *d* terms.

We conclude then that Circuit II cannot have fewer V units than Circuit I. But, as V was chosen to be any one of the variables with fewer units, we see that Circuit II must have at least as many switching units as Circuit I.

Acknowledgment

The author gratefully acknowledges the many constructive suggestions and criticisms of M. K. Haynes and R. P. Holten of the IBM Research Laboratory, Yorktown Heights, N. Y.[6]

[6] Since the completion of the paper the author has learned of the work of A. H. Sheinman, "A numerical-graphical method for synthesizing switching circuits," *Trans. AIEE*, vol. 76 (*Commun. and Electronics*, pt. I), pp. 687–689; January, 1959 and "The numerical-graphical method in the design of multiterminal switching circuits," *Trans. AIEE*, vol. 78 (*Commun. and Electronics*, pt. I), pp. 515–519; November, 1959.

Geometric Mapping of Switching Functions *

M. E. ARTHUR†

Summary—The geometric map described in this paper is a graphic representation, using vectors, of the conditions desired in the design of switching networks. This mapping technique makes the well-known techniques such as Boolean algebra, the Quine-McCluskey minimization chart, and Huffman's flow table more effective when designing optimum circuits. Its degree of flexibility in the selection of vectors is advantageous in certain problems.

Introduction

THE mathematical treatment used in designing optimum switching circuits has advanced at an ever-increasing rate since the publication of Shannon's first paper in 1938.[1,2] Switching algebra, frequently called Boolean algebra, is accepted as a useful and powerful tool, and may be used to represent functions of an infinite number of variables. In practice, however, some functions of as few as four, five, or six variables may seem unmanageable, especially to a beginner. As an aid to visualizing a Boolean expression, Veitch[3] proposed a chart in 1952, and Karnaugh[4] offered a refinement in 1953 which has to a great extent superseded the Veitch chart. The Karnaugh map is not too successful when more than four variables appear in an expression. The Geometric Map described in this paper has been used on problems involving as many as ten variables. It is intended to supplement switching algebra, not to replace it. The Geometric Map also works well with other techniques, such as the Quine-McCluskey minimization[5] and Huffman's sequential circuit synthesis.[6] It has a degree of flexibility which can be an advantage in certain problems.

Framework of Geometric Mapping

Just as the Karnaugh map has a framework of squares in which entries can be made, the geometric map has a framework of vectors, one for each variable. These vectors differ in length or direction, or both and are generally chosen to form squares or cubes. In the Geometric Map only the ends of the vectors are important (however, an exception will be discussed later). In Keister, Ritchie, and Washburn[7] and in Caldwell[8] one, two, and three variables are represented by a line, a square, and a cube, respectively. However, Keister, Ritchie, and Washburn "unfold" the cube into a planar figure equivalent to a three-variable Karnaugh map, while Caldwell adds another variable to form a four-dimensional hypercube. The authors of both books show symbols placed at the "corners" of the geometric figures, a practice continued in this paper. On a diagram of Caldwell's,[9] *all* the vectors are drawn and they are *all* the same length. Their directions were chosen to make a symmetric geometric figure. The very presence of all the vectors tends to reduce the usefulness of the figure to some extent. In the geometric map, *not all* vectors are drawn on the map, although external vectors are drawn to identify the vector with its variable.

Fig. 1 shows one of the possible frameworks for each of the two through seven variables, here shown as X_0 through X_6. If the variable X_j is assumed to have a weight 2^j, and all $\overline{X}_j$ are weighted zero, the sum of weights may be entered at the "corners" of the squares, as shown in Fig. 1. These numbers are the "condition

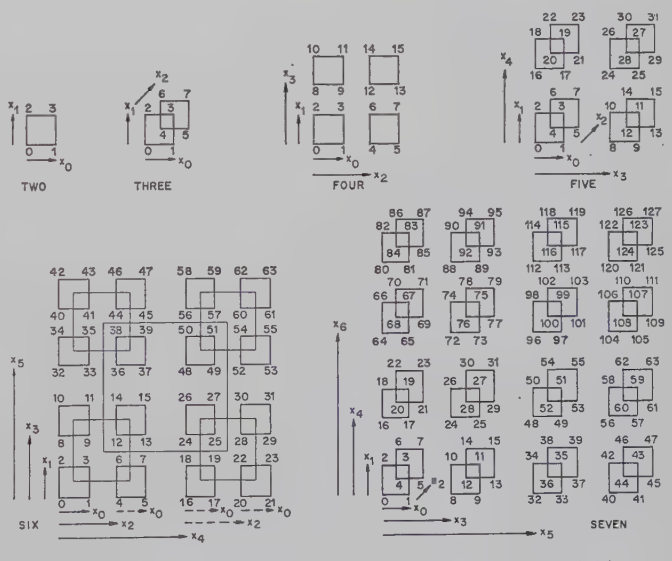


Fig. 1—Possible frameworks for two through seven variables.

* Received by the PGEC, November 21, 1960.

† Space Guidance Ctr., International Business Machines Corp., Owego, N. Y.

[1] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Trans. AIEE*, vol. 57, pp. 713–723; December, 1938.

[2] C. E. Shannon, "The synthesis of two-terminal switching circuits," *Bell Sys. Tech. J.*, vol. 28, pp. 59–98; January, 1949.

[3] E. W. Veitch, "A chart method for simplifying truth functions," *Proc. Assoc. Comp. Mach.*, Pittsburgh, Pa., pp. 127–133; May 2–3, 1952.

[4] M. Karnaugh, "The map method for synthesis of combinational logic circuits," *Trans. AIEE*, vol. 72 (*Commun. and Electronics*, pt. 1), pp. 593–599; November, 1953.

[5] E. J. McCluskey, Jr., "Algebraic Minimization and the Design of Two-Terminal Contact Networks," Ph.D. dissertation, Dept. of Elec. Engrg., Mass. Inst. Tech., Cambridge; June, 1956. The first part is published in the *Bell Sys. Tech. J.*, vol. 35, pp. 1417–1444; November, 1956.

[6] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, 275–303; March and April, 1954.

[7] W. Keister, A. E. Ritchie, and S. H. Washburn, "The Design of Switching Circuits," D. Van Nostrand Co., Inc., New York, N. Y., p. 137; 1951.

[8] S. H. Caldwell, "Switching Circuits and Logical Design," John Wiley and Sons, Inc., New York, N. Y.; 1958. See especially pp. 126–127.

[9] *Ibid.*, p. 129.

numbers" of McCluskey, Caldwell, and others. Other possible frameworks for five variables might be composed of a large cube of small squares, or of the lower half of the framework for six variables. Also, six variables may be represented on a large cube of small cubes. If desired, the intermediate and large squares may be superimposed, as on the six-variable frame; however, it is not advisable to superimpose squares because of the extra time needed to draw them and their tendency to cloud the diagram. A little practice makes them unnecessary.

In use, the actual literals or variables of the problem replace the generic variables X_j, and the framework then becomes an orderly arrangement of the 2^n input conditions for these variables. Conditions 3 and 11 in all the diagrams for more than three variables are clearly at opposite ends of the X_3 variable, as are conditions 5 and 13. Usually, other symbols are used in place of the condition numbers—the most common symbol being a dot of sufficient size to distinguish the dotted corners from the undotted ones.

The arrangement of the vertices of the squares corresponds more nearly to the Veitch chart than to the Karnaugh map, where squares of squares are used to represent an even number of variables. When cubes are used, as for the framework of an odd number of variables, there is no correspondence of position to either of these graphic symbols. The shorter vectors are mentally repeated and are shown as dotted horizontal vectors beneath the framework for six variables. Experience has shown that this repetition is unnecessary in practice.

The framework for the variables Y and Z is shown in Fig. 2(a). The symbols of the individual variables [Fig. 2(b)] on this frame assume that a two-terminal relay network or a single-output electronic switching network is being mapped. The dots indicate that the transmission is to be 1 or the output is to appear for the combination(s) of states of the variables. The undotted corners indicate that the transmission is to be 0 or that the output is not to appear. In the formation of the sums or products of these symbols, each corner of the square is considered separately. For a sum of two symbols, a dot will be placed at any corner having a dot in either or both of the symbols being added; while for a product, a dot will be placed only at corners having dots in the symbols of both factors, as shown in Fig. 2(c)–(e). This is true even in the trivial cases of a sum or a product of a variable and its inverse, or complement.

Figs. 3 and 4 are similar examples for the three variables X, Y, and Z, and the four variables W, X, Y, and Z, respectively.

It should be noted that in Figs. 2–4 a product of all n variables appears as a single dot, and a product of $n-1$ of the variables appears as two dots which are at opposite ends of the vector of the missing variable. A product of $n-2$ of the variables appears as four dots which are arranged in the "square" of the two vectors of the missing variables. For example, the symbol for $X\bar{Z}$ of Fig. 4(d) is a formalized representation of such a square, and if dots are drawn on a 4th dimensional cube, as shown in Fig. 4(f), they will form one square of the hypercube. Similarly, in a sum of all n variables, only one vertex is not dotted, and a sum of $n-1$ of the variables will have all but two vertices dotted.

The geometric map brings out "adjacencies" of the

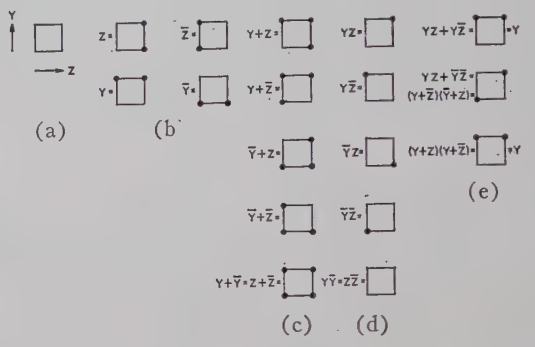


Fig. 2—Mapping of Y and Z variables. (a) Two variable framework for Y and Z. (b) Symbols of the individual variables. (c) Symbols of sums of the variables. (d) Symbols of products of the variables. (e) Symbols of sums of products and products of sums.

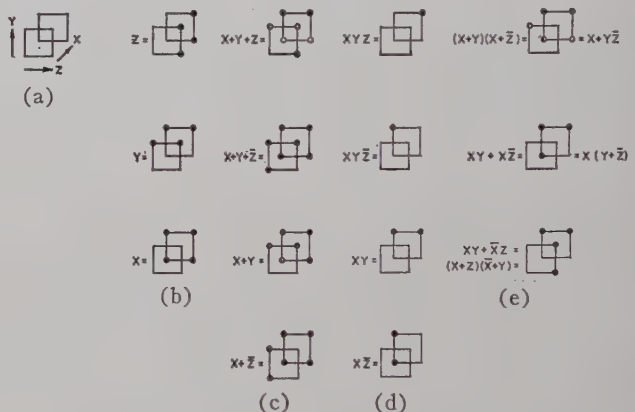


Fig. 3—Mapping of XYZ variables. (a) $F(XYZ)$ frame. (b) Examples of individual symbols. (c) Symbols of sums, (d) Symbols of products. (e) Symbols of mixed expressions.

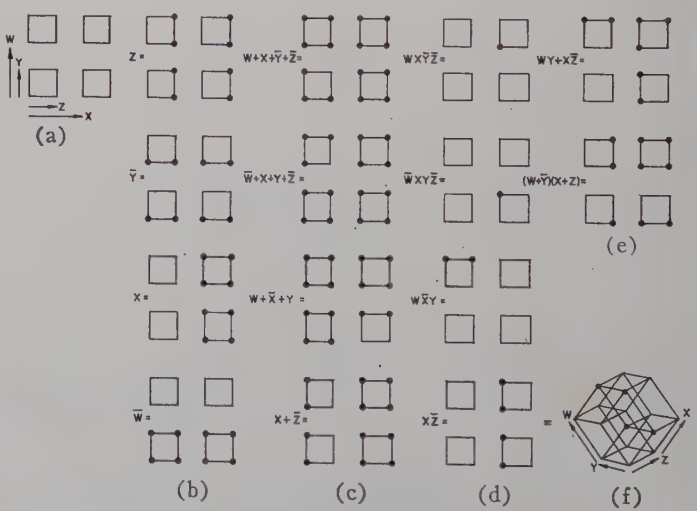


Fig. 4—Mapping of $WXYZ$ variables. (a) $f(WXYZ)$ frame.

corresponding parts of sections of the total map. Any vertex on an n variable map is adjacent to n other vertices. In the minimization of Boolean expressions we are constantly looking for adjacent points or sets of points, all of which represent a common output condition. For example, consider the function $AB+\overline{A}C+BC$, shown in Fig. 5. Here, the lower-case letters are placed by the dots, so that the contribution of each term can be examined separately. Since one variable is missing from each term, the small letters must appear in pairs at each end of the proper vector. We see that a appears alone at the vertex corresponding to $AB\overline{C}$, and b at the $\overline{A}BC$ vertex. However, c does not appear alone. Consequently, the term represented by c is redundant. Fig. 6 is a similar example of the dual of the preceding example $(AB+\overline{A}C+BC)$. Dual refers to the interchanging of multiplication and addition with no change of the literals. In Fig. 6(a), there should be an output for only those input conditions represented by the corners of the cube at which a, b, and c all appear, since any one missing contributes a zero to the product. These corners are dotted. Since Fig. 6 is treating a product-of-sums expression, the failure of c to appear alone is not sufficient to label the factor that c represents redundant. It is redundant, but the redundancy is shown by the fact that ab never appears except as part of the entry abc. This is shown more clearly by the inverse symbol of Fig. 6(b). The bar over the diagram is to be interpreted as meaning that Fig. 6(b) is the inverse of Fig. 6(a). In Fig. 6(b), the a's indicate the conditions for which factor a prevents an output, and similarly for b and c. To form this inverted symbol, each corner of the cube of Fig. 6(a) is examined and all the missing lower case letters are entered on the corresponding corner of Fig. 6(b). Now c never appears alone in Fig. 6(b); hence, it is redundant. But for c to appear alone in Fig. 6(b), ab would have to appear in Fig. 6(a).

Now consider the function $AB+\overline{A}C+\overline{B}C$ mapped in Fig. 7. The redundancy in this case is somewhat different from the preceding cases. The lower-case letters reveal no term that can be dropped completely. However, conditions 4–7, shown in the three-variable framework in Fig. 1, are all present and may be combined into the single literal C. If conditions 3 and 7 are separated, the result is $AB\overline{C}+C$. Since there is no reason for this separation, conditions 3 and 7, as a pair, yield the result $AB+C$.

Similarly, $(A+B)(\overline{A}+C)(\overline{B}+C)$ may be mapped as shown in Fig. 8 (note the change in vector assignments). Again, the corners at which abc appear are dotted, since these are the only conditions of the variables for which an output can occur. The dots indicate that C and either A (conditions 6 and 7) or B (conditions 3 and 7) or both (condition 7) must be on for the output to occur. This expression reduces to $C(A+B)$.

As a four-variable example, consider $AB+\overline{A}C\overline{D}+\overline{B}D+CD$, mapped in Fig. 9(a). Since conditions 3 and 7 are present, the second term may be changed from $\overline{A}C\overline{D}$ to $\overline{A}C$. By making this change on the map [adding the b's in parentheses as shown in Fig. 9(b)], the d term (CD) is totally redundant; hence, it may be dropped.

On the map of the function $(A+B)(\overline{A}+C)(\overline{B}+D)(C+E)(\overline{D}+E)$ shown in Fig. 10(a) some time would be required to note that $abce$ never appears except as part of $abcde$. The inverse map, Fig. 10(b), shows clearly that a, b, c, and e appear alone, while d never appears alone.

The preceding examples could have been accomplished by Boolean algebra alone. They were chosen to show how geometric mapping speeds the Boolean operation.

Treatment of k Cells

A "k cell" is defined as the 2^k vertices of a geometric map which include all the possible combinations of

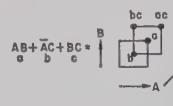


Fig. 5—Mapping of function $AB+\overline{A}C+BC$.

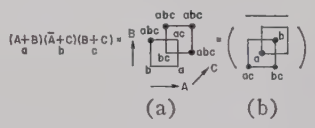


Fig. 6—Mapping of function $(A+B)(\overline{A}+C)(B+C)$.

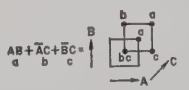


Fig. 7—Mapping of function $AB+\overline{A}C+\overline{B}C$.

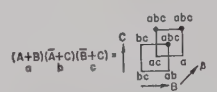


Fig. 8—Mapping of function $(A+B)(A+C)(\overline{B}+C)$.

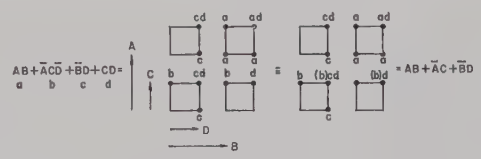


Fig. 9—Mapping of function $AB+\overline{A}C\overline{D}+\overline{B}D+CD$.

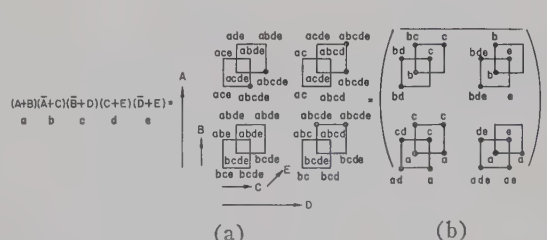


Fig. 10—Mapping of function $(A+B)(\overline{A}+C)(\overline{B}+D)(C+E)(\overline{D}+E)$.

some k variables. Thus, a zero-cell is a single vertex; a one-cell is any pair of vertices at opposite ends of one vector, etc. As examples, each of the lower-case letters in Figs. 5, 6(b), and 7 are examples of one-cells. In Fig. 9, the lower-case letters, a, c, and d are two-cells, while b starts as a one-cell but becomes a two-cell to make the d two-cell redundant. In Fig. 8, each letter covers two two-cells with a one-cell in common; hence, they do not complete three-cells. In Fig. 10(a), each lower-case letter covers two four-cells with a three-cell in common. It will be instructive to locate them in the diagram.

Let us examine the function $f(WXYZ) = (2, 3, 4, 6, 8, 10, 14, 15)$ shown in Fig. 11. The two-cell of a's looks attractive, but is unnecessary, since it is completely covered by the four one-cells marked b, c, d, and e. The same thing is shown by the Quine-McCluskey prime-implicant chart: prime implicants b, c, d, and e are primary basis rows (Caldwell's nomenclature) and a is redundant. In this case, the prime-implicant chart did not indicate anything more than did the geometric map. In more complex problems, the chart will emphasize the cells that are covered by other cells in this manner.

Symmetric Circuits

McCluskey arranges the binary equivalents of the decimal condition numbers in order of increasing index of the binary numbers. The index is the number of "ones" in the binary number, regardless of where these "ones" stand. Fig. 12 shows dotted lines superimposed on frames for two, three, four, and five variables. By redirection of the long vectors, these "index lines" may be drawn as straight lines on frames for more than three variables. However, the more formal forms for four and more variables may be used with practice. These index lines have real value in symmetric relay problems. If all the input conditions on each index line have the same output condition, considering each index line separately, the problem is one of a symmetric circuit. It is suggested that the reader demonstrate to himself, by interchanging the variables in any manner and reassigning the condition numbers accordingly, that exactly the same condition numbers will appear on each index line. This property is related to a characteristic of symmetric functions in that it is not *which* relay is operated, but *how many* of them are operated.

Let us examine the function $\overline{W}\overline{X}YZ+\overline{W}X\overline{Y}Z+\overline{W}XY\overline{Z}+WXYZ$, mapped in Fig. 13(a)–(c), showing three choices of vector assignments. If we rotate the small individual squares of Fig. 13(a) clockwise 90°, we get Fig. 13(d); or rotating the small squares of Fig. 13(b) counterclockwise 90°, we get Fig. 13(e); while reversing the W vector of Fig. 13(c) yields Fig. 13(f). Fig. 13(d)–(f) has three somewhat different vector assignments, but the dots all fall on the number 3 index line, so this function is a symmetric three of $\overline{W}$, X, Y, and Z, or any permutation of these four variables, although not symmetric with respect to W, X, Y, and Z.

Let us examine the map of Fig. 14(a) for symmetry. Only the index lines 0 and 5 are complete. Notice that the pattern of dots on the lower left and upper right cubes is exactly the same; furthermore, there is sym-

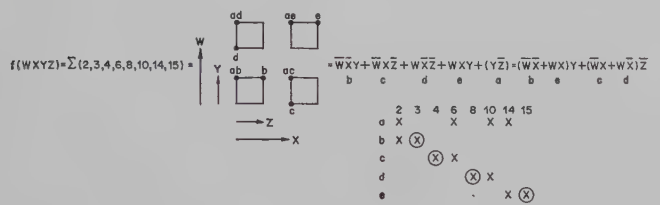


Fig. 11—Mapping of function $WXYZ = \Sigma(2, 3, 4, 6, 7, 10, 14, 15)$.

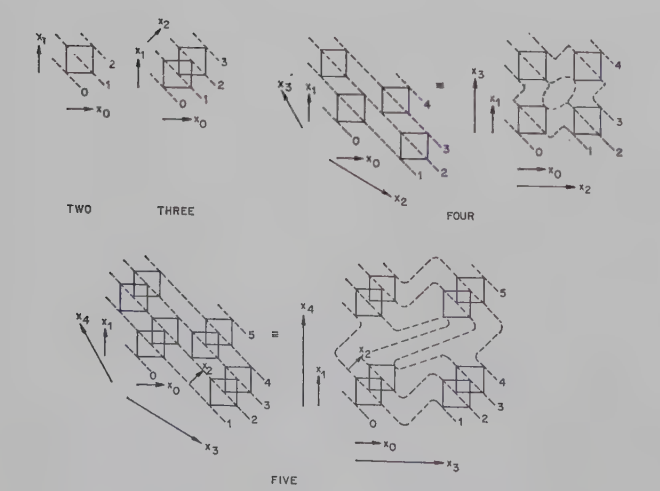


Fig. 12—Index lines for two, three, four, and five variables.

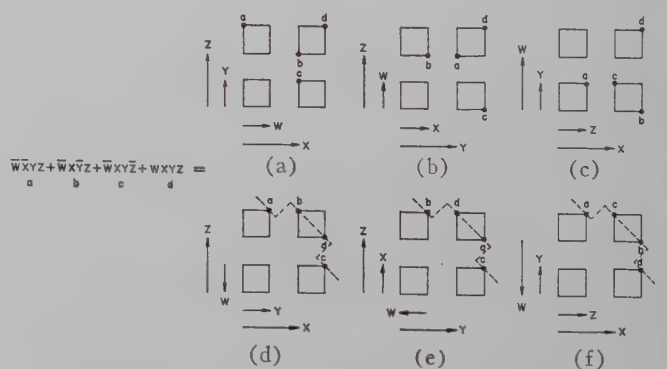


Fig. 13—Symmetry of function $\overline{W}XYZ+\overline{W}X=Z+\overline{W}XY\overline{Z}+WXYZ$.

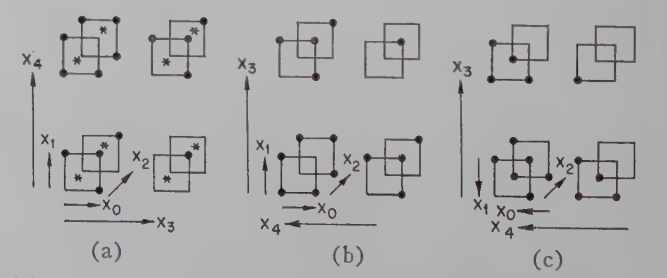


Fig. 14—Symmetry of five variable frames. (a) $f[X_0X_1X_2X_3X_4]$. (b) $f[X_0X_1X_2\overline{X}_4X_3]$. (c) $f\overline{X}_0\overline{X}_1X_2\overline{X}_4X_3] = S_{12}\overline{X}_0\overline{X}_1X_2X_3\overline{X}_4]$.

metry on all cubes with respect to parallel diagonals of the cubes. The end of these diagonals are marked with stars. Rotating the large square 90° counterclockwise, without revolving the cubes, yields Fig. 14(b), which is now symmetric with respect to X_3 and $\overline{X}_4$. Rotating each of the small cubes of Fig. 14(b) 180° around the X_2 vector, yields Fig. 14(c). Comparing Fig. 14(c) with the five variable frames of Fig. 12 shows that this function is $S_{1,2}$ ($\overline{X}_4$, X_3, X_2, $\overline{X}_1$, $\overline{X}_0$). Had the large squares of Fig. 14(a) been rotated 90° clockwise, a similar set of steps would have led to $S_{3,4}$ (X_4, $\overline{X}_3$, $\overline{X}_2$, X_1, X_0), which is another form of the same expression.

Concept of Patterns

The concept of patterns, when enlarged to mean the relative positions of all the dots on a symbol, is an important one in connection with geometric mapping. Thus, the maps of Fig. 13 show four different positions of one pattern, and those of Fig. 14 show three positions of another pattern. On any map for n variables there are $n!$ possible assignments of the variables to the vectors. For every such permutation of the variables there are also 2^n inversions of the variables, shown in Figs. 13 and 14, which are obtained by reversing some vectors. Hence there are $2^n \times n!$ maps for any function of n variables. Symmetry, either partial or total, reduces the number of positions any one pattern may assume [note Fig. 13(d)–(f)]. This being a 4-variable example, it could have 384 maps, but due to symmetry these maps will appear in only 16 different positions, 4 of which are shown. Figs. 5 and 6 display three positions of another pattern, which can appear in 24 positions. These patterns correspond to Caldwell's classes of Boolean functions and to Slepian's[10] symmetry types. By making full use of patterns, only 6 circuit types or classes need to be known to satisfy all the 16 specific functions of 2 variables, only 22 types for the 256 functions of 3 variables, and only 402 types for the 65,536 functions of 4 variables.

The author made a list of the 402 circuit types for 4 variables only to discover it was not universally useful because of what we may call ground rules. Ground rules may exist because of efforts at standardization, the necessity to use what is available, the need to use items or circuits that will "fail safe," or for any number of other reasons. Standardization may cause us to use a diode AND gate which can drive a diode OR gate in an electronic circuit; however, the OR gate cannot directly drive another AND gate of the same kind. This ground rule would prohibit instrumenting the function $(A+B)(C+D)$ using two OR gates to drive an AND gate. Using whatever relays are available might cause one to use solenoid type ac relays, whose release time is less than their transit time. This ground rule would require careful design of any circuit in which a normal transfer contact was used in the hold path of another such relay. In the design of circuits to control the warning lights at a railroad grade crossing, it is better for the lamps to flash needlessly than for them to fail when a train approaches.

Let us assume that we must design a relay circuit to control relay Y according to the map of Fig. 11, using the solenoid type ac relays. A "first try" circuit might be the one shown in Fig. 15(a). In this circuit the coil of relay Y will be energized by the one-cells c and d, that is, either W or X operated, but not both. Now if Z is operated, Y is de-energized, and if Z is again released, Y will again be energized; however, if either W or X change state, Y will be de-energized before the required hold path is established. Knowing of the a two-cell of Fig. 11, we might make the circuit usable by adding another Y or Z contact. A more economical circuit results if we note that $\overline{W}\overline{X}+WX$ is the complement of $\overline{W}X+W\overline{X}$ and use the Boolean identity $VY+\overline{V}\overline{Z}=(V+\overline{Z})(\overline{V}+Y)$ where $V=\overline{W}\overline{X}+WX$, as shown in Fig. 15(b). The redundant YZ term is included by this rearrangement. In any example where continuity must be maintained, it may be shown by adding dotted lines from dot to adjacent dot as in Fig. 15(c). Again, experience has shown that if one is aware of this necessity, the dotted lines are not needed.

Another group of symbols is shown on the same map in Fig. 15(d). The numbers in this case might be the "state numbers" of a Huffman[6] flow table. For the relay Y a number in a circle represents a stable state, and the number not in a circle represents an unstable state. For instance, in states ①, ②, ④, ⑧, etc., Y is unoperated and de-energized (stable states) and in states ⑤, ⑥, ③, ⑦, etc., Y is energized and operated (again, stable states). Note that with X on alone (state 3), Y is energized, but unoperated. Assuming that this state persists long enough for Y to respond, Y will operate, changing the point of our interest from 3 to ③, where Y is energized and operated. Should Z now also become operated, our

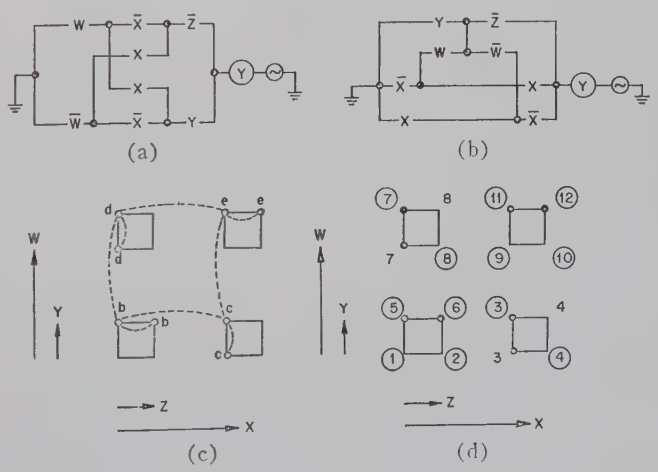


Fig. 15—Relay circuit using ac solenoid-type relays and corresponding state numbers.

[10] D. Slepian, "On the number of symmetry types of Boolean functions of n variables," *Canad. J. Math.*, vol. 5, pp. 185–193; February, 1953.

interest would move from ③ to 4, where Y is operated but de-energized (an unstable state). Y will then release, moving our interest to ④. As shown in Fig. 15(d), when W and X are both off or both on Y has only stable states; however, with either W or X on alone Y may have unstable states. This map gives us a feel for the action of the circuit of Fig. 15(b).

A Sequential Example

Let us examine the use of the geometric map in the design of an electronic circuit to control a stepping motor. This stepping motor has a permanent magnet rotor and two centertapped stator windings which are mechanically arranged at right angles to each other. There is no mechanical detent, the rotor being moved to or held at a given position by applying power to one or the other half of one or both stator windings. For the present example power will be applied to both stator windings to take advantage of the 40 per cent increase in output torque. The two inputs, F and R (shown in Fig. 16), never occur together or close together. Each pulse on line F advances the stepping motor one step of 90°, clockwise, and each pulse on line R moves the stepping motor one step of 90°, counterclockwise. To avoid "ground rule" problems, it will be assumed that the pulses are adequate to operate the flip-flops, used for internal memory, even after passing through the AND and OR gates. The AND gates will be symbolized by triangles, the OR gates by semicircles, the flip-flops by rectangles identified by FF, and the power drivers for the motor by rectangles identified by D.

The first step in the formation of the primitive flow table[6] is to note that for each of the four possible positions of the stepping motor there exists a stable internal state for each of the following three input conditions: 1) no input pulses, 2) a forward input pulse, and 3) a reverse input pulse. In Fig. 17(a), these stable-state numbers are arranged in arbitrary order and the changes of input, ruled out by the statement that the pulses never occur together or close together, have been filled in with dashes. Fig. 17(b) is the same table as Fig. 17(a) with the required changes of the secondaries to produce the changes of the output (Z column) described above, and with the 11 column dropped because it is not required. As it appears the table of Fig. 17(b) would require four internal flip-flops to separate the twelve stable internal states. Rows which have no conflicting entries may be "merged," providing circled entries retain their circles, and dashes do not conflict with any other entry. The table of Fig. 17(b) may, therefore, be shortened to that of Fig. 18(a) for which only three flip-flops are required. Before drawing the geometric map, it is well to examine the sequence which the rows of Fig. 18(a) must follow to avoid critical races or an excessive number of secondary states. By referring to the letters to the right of the Z column in Fig. 18(a), a figure such as 18(b) can be drawn. Each loop in such a figure as this should have an even number of nodes, extra node(s) being added if needed. Practical problems occur in which it is possible to go from one secondary state to a second by passing through a third state, which also leads to the second state. This requires two secondary changes possibly in a given order but may be acceptable at times. Since this example has two inputs and three secondaries, or internal elements, a five-variable frame is needed. A cube of squares is used with the inputs on the squares and with the secondaries on the cube as shown in Fig. 18(c), since this emphasizes the difference between the two kinds of variables. In the first step of preparing the geo-

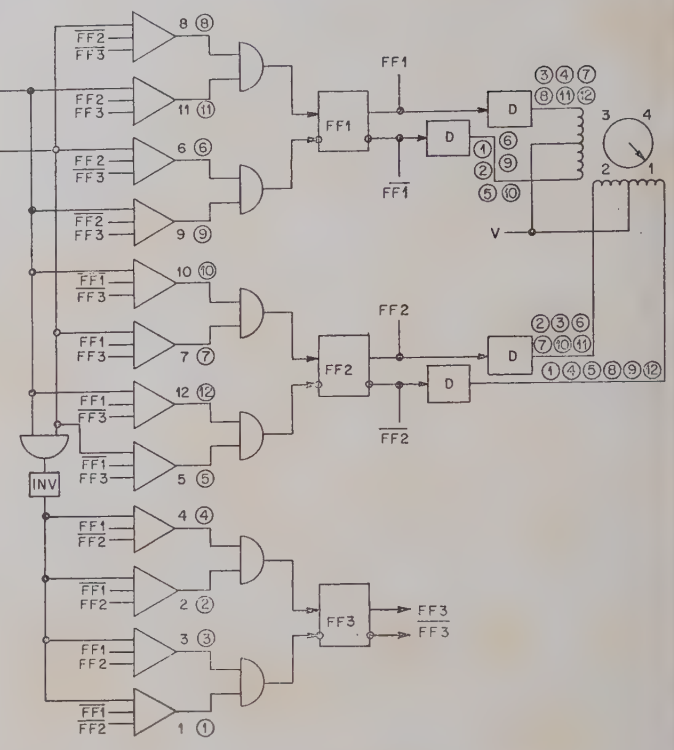


Fig. 16—Circuit for stepping motor.

FR					FR			
00	01	10	11	Z	00	01	10	Z
①			—	1	①	8	10	1
②			—	2	②	5	11	2
③			—	3	③	6	12	3
④			—	4	④	7	9	4
	⑤	—	—	1	1	⑤	—	1
	⑥	—	—	2	2	⑥	—	2
	⑦	—	—	3	3	⑦	—	3
	⑧	—	—	4	4	⑧	—	4
	—	⑨	—	1	1	—	⑨	1
	—	⑩	—	2	2	—	⑩	2
	—	⑪	—	3	3	—	⑪	3
	—	⑫	—	4	4	—	⑫	4
	(a)					(b)		

Fig. 17—Primitive flow tables.

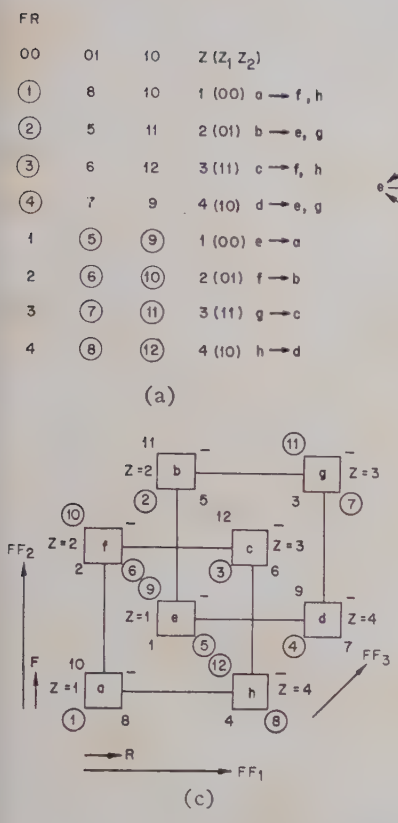


Fig. 18—Geometric mapping of a circuit to control a stepping motor.

metric map, the letters of Fig. 18(b) are entered in the squares so that only one flip-flop changes state for each change of inputs. Caldwell[8] shows how this is always possible but may require added secondaries. Then the state numbers of Fig. 18(a) are entered at the proper corners of the squares. The dashes at the upper right corners of the squares are the same as the 11 column of Fig. 17(a) and are invalid conditions (these dashes will never occur, according to the statement of the problem). We may use these conditions where the dashes appear as we please to simplify the circuit, if we are willing to take the calculated risk should any of these inputs occur. From Fig. 18(c) we must now extract the controlling conditions for each of the flip-flops and the output. Since a flip-flop once set to a given state by a pulse on one input will remain in that state until reset by a pulse on the other input, we do not have the problem of maintaining continuity, as in Fig. 15. As an aid to extracting these controls, Fig. 18(b) may be augmented to show which flip-flop changes state when going from letter to letter, as in Fig. 18(d). In Fig. 19(a)–(c) dots indicate that the

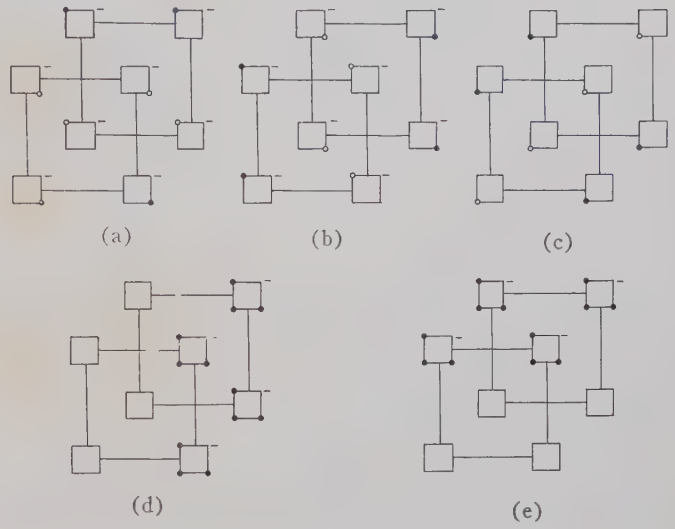


Fig. 19—Mapping of trigger states for stepping motor circuit. (a) $f[T_1]$. (b) $f[T_2]$. (c) $f[T_3]$. (d) $f[Z_1]=T_1$. (e) $f[Z_2]=T_2$.

flip-flop is to be set and circles indicate that the flip-flop is to be reset; where no change is required for a given flip-flop, no entry is made. The only dashes shown are for those invalid conditions used to simplify the circuits.

Before their geometric maps are drawn, the outputs, designated by decimal numbers in Figs. 17 and 18(a), must be converted to a cyclic-binary code to be compatible with the stepping motor. One possible code is shown in Z_1Z_2 column of Fig. 18(a). If we assume that the response of the stepping motor will be adequately rapid when controlled by only the flip-flops, the maps of Fig. 18(d) and (e) can be used for the outputs. This is a reasonable assumption since the flip-flops should respond in microseconds while the stepping motor will require milliseconds. In the circuits for FF1 and FF2, Fig. 16, the "invalid" conditions are used, thereby saving eight diodes in the AND gates. The state numbers for which each AND gate opens is shown by its output lead, and the state numbers for which each driver allows current through the motor winding is shown by its output line.

Conclusion

Although problems involving 12 variables have been mapped, the examples presented here have been limited to fewer than 6 variables to introduce the subject. Having once mastered this technique on the fewer variables the geometric map seems to gain power with greater numbers of variables.

Bibliography on Switching Circuits and Logical Algebra*

PER A. HOLST†

Summary—This bibliography, which covers material published through 1958, contains nearly 700 references to articles, books, seminars, and other bibliographies pertaining to the theory of switching circuits and logical algebra. A relatively large number of the references are to foreign works, of which the Russian contribution is by far the greatest. An effort has been made to give exact bibliographical references; wherever possible, the titles of the foreign items have been listed in their original language, together with an English translation. Appended to the chronological bibliography are a list of books, a list of other bibliographies, an author index, a subject index, a list of periodicals, and a list of frequently used Russian abbreviations.

Introduction

THE nearly 700 references in the bibliography cover the time period through 1958. They are arranged alphabetically by author for each year from 1938 through 1958, and are preceded by a chronological listing of the "prehistoric" papers published prior to 1938. The year 1938 was selected because of the important works presented at that time, such as the well-known paper by Claude E. Shannon [33], the English translation of a number of Japanese papers by A. Nakasima [19], [20], [25], [30]–[32], the dissertation at the Lomonosov State University by the Russian, V. I. Shestakov [35], and a variety of German, Austrian, and other papers.

During the succeeding years, a rapidly increasing number of works have appeared in the fields of switching circuits and logical algebra, in the areas of both theory and application. Except for a relatively few scattered papers from Europe and the far East, the major contributions have come from American and Russian works. These are often parallel or complementary due to linguistic and other barriers preventing the complete exchange of information. Only in the last few years has any considerable amount of translational service been available to improve this situation. Since a major part of the non-English references in this bibliography deals with Russian and other Eastern European works on switching circuits and logical algebra, it is hoped that this work may be a contribution to the reduction of the existing barriers.

The references have been listed with name of author or authors, title (in the original language whenever possible, either directly or in a transliterated form, with an English translation of the title), and the bibliographical information pertaining to periodical, volume, number, pages, and date. In a few cases, the available information has been considered incomplete or unreliable, but in the interest of completeness, nothing known has been omitted. Since the bibliography was compiled during a short period of time, only a few cross checks could be made; thus, many significant works may have been overlooked.[1] References to works on the applications of switching circuits and logical algebra have been included when found to be of sufficient general interest.

Titles and names in Russian have been given in a transliterated form, corresponding to the standard English transliteration alphabet, with the exception of the letters "ю" and "я," which have been transliterated "ju" and "ja," respectively, instead of the usual "yu" and "ya," mainly for the spelling of names, *e.g.*, Jakovlev instead of Yakovlev. Also, the letter "й" has been replaced by "j."

At the end of the bibliography a separate list of books is given, together with a list of additional bibliographies giving references in the fields of switching circuits and logical algebra. An author index and a subject index are also included, together with a list of the periodicals mentioned in the references and a list of frequently used Russian abbreviations.

Acknowledgment

This bibliography was compiled during the fall term of the academic year 1960–1961 at Massachusetts Institute of Technology, Cambridge, Mass., during a stay made possible by the Chr. Michelsens Institute, Bergen, Norway, and by a scholarship granted by the Royal Norwegian Council for Technical and Industrial Research (N.T.N.F.), Oslo, Norway, for which the author is very grateful. Also, the author is grateful for permission to use the research facilities at the libraries of M.I.T. and for the staff's cooperative assistance.

Prior to 1938

[1] Kolbe, "Das Entwerfen von Umschalt-Einrichtungen" (Outline of switching devices), *Z. Elektrotech.*, vol. 11, no. 1, p. 5; 1893. (In German.)
[2] Boda, M., "Die Stromlaufformeln und ihre Anwendung zur Schaltung Siemensscher Blockwerke [Versuch einer Schaltungstheorie Siemensscher Blockapparate]" (Current path equations and their application in controlling Siemens' railroad switch systems [Experiments with a contact network theory for the Siemens' railroad switch apparatus]), *Z. Österr. Ing. Arch. Verrein*, vol. 49, p. 620; 1897. (In German.)
[3] Boda, M., "Die Schaltungstheorie der Blockwerke" (Contact network theory for railroad switching systems), *Org. Fortschr. Eisenbahnwes.*, vol. 35, nos. 1–7; 1898. (In German.)
[4] Edler, R., "Analytische und Graphische Bestimmung der Schaltungen im elektrischen Signalwesen" (Analytical and

* Received by the PGEC, April 12, 1961.
† Appl. Phys. Dept., Chr. Michelsens Institute, Bergen, Norway.

[1] The author would appreciate communication from readers for a future revision of this bibliography.

graphical solution in determining switching networks in electric signaling systems), *Mitt. Technol. Gewerbemuseums Wien;* 1900. (In German.)
[5] Walzel, I., "Die Schaltung Siemensscher Blockwerke" (Contact networks for the Siemens' railroad switches), *Org. Forschr. Eisenbahnwes.*, vol. 37, p. 158; 1900. (In German.)
[6] Edler, R., "Eine einfache Methode zur Bestimmung von Schaltungen [Schaltungs-theorie]" (A simple method for determining switching networks [switching theory]), *Mitt. Technol. Gewerbemuseums Wien;* 1903. (In German.)
[7] Edler, R., "Über den Entwurf von Schaltungen und Schaltapparaten [Schaltungstheorie]" (Sketch of contact networks and switching devices [switching theory]), *Z. Elektrotech.*, vol. 21, nos. 31, 32, pp. 499, 465; 1903. (In German.)
[8] Torres-Quevedo, L., "Essais sur l'automatique. Sa definition. Etendue théorique de ses applications" (Essay on automatism. Its definition. Theoretical extent of its applications), *Rev. gén. sci.*, November 15, 1915. (In French.)
[9] Kutti, A., "O graficheskom izobrazhenii rabochego rezhima skhem" (Graphical representation of circuit performance conditions.), *Trudy Leningrad Eksper. Elektrotekh. Labor.*, no. 8, p. 11; 1928. (In Russian.)
[10] Tsimbalistyj, M., "K voprosu o ratsionalnom sostavlenii relejnykh skhem" (On the question of simplification in the construction of relay circuits), *Trudy Leningrad Eksper. Elektrotekh. Labor*, no. 8, p. 19; 1928. (In Russian.)
[11] Edler, R., "Wahlschalter [Umschalter für 3 oder 4 Stromwege]" (Selector switch [Switch for 3 or 4 current paths]), *Elektrotech. u. Maschinenbau*, vol. 49, no. 46, pp. 848–851, November, 1931. (In German.)
[12] Mays, W., "The first circuit for an electrical logic-machine," *Science*, vol. 118, no. 3062, pp. 281–282; 1933.
[13] Winkel, E., "The use of interlinking and time diagrams for simplifying the study of complicated circuit diagrams," *Ericsson Tech.*, no. 6; 1934.
[14] Bader, W., "Wechselschaltungen, Theorie und Anwendungen" (Theory and applications of multipoint switching systems), *Arch Elektrotech.*, vol. 29, no. 2, pp. 107–119; February, 1935. (In German.)
[15] Edler, R., "Packetschalter. Ein Beitrag zur Schaltlehre und zur Systematik der Packetschalter" (Packet switch. Contribution to switching theory and systematism of packet switches), *Arch. Elektrotech.*, vol. 29, pp. 531–555; August, 1935. (In German.)
[16] Stone, M. H., "Postulates for Boolean algebras and generalized Boolean algebras," *Am. J. Math.*, vol. 57, pp. 703–732; 1935.
[17] Winkel, E., "Der Verkettungs- und der Schaltzeitplan als Hilfsmittel zur Darstellung der Schaltvorgänge in der Selbstanschlusstechnik" (Interconnection and switching time diagrams as aids in the description of switching courses in the automatic connecting technique), *Z. Fernmeldetech.*, no. 10; 1935. (In German.)
[18] McKinsey, J. C. C., "On Boolean functions of many variables," *Trans. Am. Math. Soc.*, vol. 40, pp. 343–362; 1936.
[19] Nakasima, A., "Theory of relay circuits," *Nippon Elec. Commun. Engrg.*, no. 3, pp. 197–226; May, 1936.
[20] Nakasima, A., and M. Hanzawa, "Theory of equivalent transformation of simple partial paths in relay circuits." *J. Inst. Elec. Commun. Engrs. Japan*, no. 165, December, 1936; no. 167, February, 1937. (Original in Japanese.) *Nippon Elec. Commun. Engrg.*, no. 9, pp. 32–39; February, 1938. (English transl.)
[21] Pankajam, S., "On symmetric functions of n elements in a Boolean algebra," *J. Indian Math. Soc.*, vol. 2, no. 5, pp. 198–210; 1936–1937.
[22] Plechl, O., "Fernüberwachung und Fernsteuerung von Schalternbedienter Betriebstellen" (Remote monitoring and control of electric switches), *Elektrotech. u. Maschinenbau*, vol. 54, no. 35, pp. 421–426; September, 1936. (In German.)
[23] Quine, W. V., "Logic based on inclusion and abstraction," *J. Symbolic Logic*, vol. 1, pp. 145–152; 1936.
[24] Stone, M. H., "Representations of Boolean algebras," *Trans. Am. Math. Soc.*, vol. 40, pp. 37–111; July, 1936.
[25] Nakasima, A., "Algebraic expressions relative to simple partial paths in the relay circuits," *J. Inst. Elec. Commun. Engrs. Japan*, no. 173; August, 1937. (Original in Japanese.) *Nippon Elec. Commun. Engrg.*, no. 12, pp. 310–314; September, 1938. (English transl.)

Books and Book Reviews:

[26] Edler, R., "Entwurf von Schaltungen und Schaltapparaten [Schaltungstheorie]" (Outline of contact networks and switching devices [switching theory]), Dr. Max Jänecke Verlagsbuchhandel, Hannover; 1905. "Schaltlehre [Wege zum Schaltplan]" (Switching theory [ways to the switching systems]), Dr. Max Jänecke Verlagsbuchhandel, Leipzig, 2nd ed.; 1927. "Schaltlehre [Entwicklung und Anwendungen]" (Switching theory [development and applications]), Franz Deuticke, Wien, 3rd ed.; 1952. (In German.)
[27] Lischke, R., "Schaltlehre—Anleitung zur Ansmittlung von Schaltungen elektrischer Einrichtungen" (Switching theory—guidance in development of switching networks in electrical systems), Verlag Hachmeister und Thal, Leipzig, 1911; 3rd ed., 1921. (In German.)
[28] Schwaiger, "Besprechung des Buches von R. Edler" (Review of R. Edler's book), *Elektrotech. Z.*, vol. 49, no. 10, p. 414.

1938

[29] Livshits, N., "Teoreticheskie osnovy rascheta i konstruirovanija apparatury teleupravlenija" (Theoretical basis for design and construction of remote control apparatus), in "Relejnye Soedinenija" (Relay Connections), Izd. ONTI; 1938. (In Russian.)
[30] Nakasima, A., "The theory of four-terminal passive networks in relay circuits," *J. Inst. Elec. Commun. Engrs. Japan.*, no. 179; April, 1937. (Original in Japanese.) *Nippon Elec. Commun. Engrg.*, no. 10, pp. 178–179; April, 1938. (English abstract.)
[31] Nakasima, A., "The theory of two-point impedance of passive networks in the relay circuit," *Nippon Elec. Commun. Engrg.*, no. 13, pp. 405–412, November, 1938. *J. Inst. Elec. Commun. Engrs. Japan.* (Original in Japanese.)
[32] Nakasima, A., "The transfer impedance of four-terminal passive networks in the relay circuits," *J. Inst. Elec. Commun. Engrs. Japan.*, no. 179; February, 1938. (Original in Japanese.) *Nippon Elec. Commun. Engrg.*, no. 14, pp. 459–466, December, 1938. (English excerpt.)
[33] Shannon, C. E., "A symbolic analysis of relay and switching circuits," *Trans. AIEE*, vol. 57, pp. 713–723; 1938. Dissertation, Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 69 pp.; 1940.

Dissertations:

[34] Ritter, A., "Beiträge zur Schaltlehre" (Contributions to the switching theory), Technische Hochschule, Wien; 1938. (In German.)
[35] Shestakov, V. I., "Nekotorye mathematicheskie metody konstruirovanija i uproshchenija dvukpoljusnykh elektricheskikh skhem klassa A" (Some mathematical methods for construction and simplification of two-terminal electrical networks of class A), Lomonosov State University, Moscow; 1938. (In Russian.)

1939

[36] Piesch, H., "Begriffe der allgemeinen Schaltungstechnik" (Principles of the general switching technique), *Arch. Elektrotech.*, vol. 33, no. 10, pp. 672–686; October, 1939. (In German.)
[37] Piesch, H., "Über die Vereinfachung von allgemeinen Schaltungen" (On the simplification of general switching networks), *Arch. Elektrotech.*, vol. 33, no. 11, pp. 733–746; November, 1939. (In German.)
[38] Rozenberg, V., "Nekotorye svojstva relejnogo nabora" (Some properties of relay systems), *Avtomat. i Telemekh.*, no. 1, pp. 37–48; 1939. (In Russian.)
[39] Winkel, E., "Die Darstellung der Schaltvorgänge in einer Relais-wählanlage nach dem Schaltzeitplan" (Description of switching courses in a relay exchange system by means of switching time diagrams), *Z. Fernmeldetech.*, no. 8–9; 1939. (In German.)

1940

[40] Pólya, G., "Sur les types des propositions composées" (On the types of combinational propositions), *J. Symbolic Logic*, vol. 5, no. 3, pp. 98–103; 1940. (In French.)
[41] Rozenberg, V., "Zadacha o blokirovke i preobrazovanii kontaktnykh grupp" (The problem of interlocking and transforming of contact groups), *Avtomat. i Telemekh.*, no. 1, pp. 47–54; 1940. (In Russian.)

1941

[42] Ritter, A., "Verfahren zur Entwicklung von Kehrschaltungen" (Methods of developing contact network inversions), *Arch. Elektrotech.;* 1941. (In German.)
[43] Shestakov, V. I., "Algebra dvukpoljusnykh skhem postroennykh iskljuchitel'no iz dvukpoljusnikov (Algebra A-skhem)" [Algebra of two-terminal networks constructed exclusively of two-terminal elements (Algebra of A-networks)], *Zhur. Tekh.*

Fiz., vol. 11, no. 6, pp. 532–549; 1941. *Avtomat. i Telemekh.*, no. 2, p. 15; 1941. (In Russian.)
[44] Shimbel, A., "Application of matrix algebra to communication nets," *Bull. Math. Biophys.*, vol. 13, pp. 165–178; 1941.

1942

[45] Lefshetz, S., "Algebraic topology," *Am. Math. Soc. Colloq. Publ.*, vol. 27; 1942.
[46] Riordan, J., and C. E. Shannon, "The number of two-terminal series-parallel networks," *J. Math. Phys.*, vol. 21, no. 2, pp. 83–93; 1942.
[47] Windmüller, A., "Switching diagrams for electric industrial plants and their practical use," *Brown Boveri Co. Nachrichten*, vol. 29, pp. 41–47; April–September, 1942. (Original in German.)

Dissertations:

[48] Joel, A. E., "Development of Relay and Switch Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge; 1942.

1943

[49] Vakhnin, M., A. Bryleev, and M. Vlodavskij, *et al.*, "Ustrojstvo S. Ts. B. i ikh soderzhanie" (The working principles of the S. Ts. B. and the maintenance), *Tranzheldorizdat*, p. 266; 1943. (In Russian.)

Dissertations:

[50] Plechl, O., "Die Kombinatorik der Strompfade elektromechanischer Schaltungen" (Combinations of current paths in electromechanical switching circuits), Technische Hochschule, Wien; 1943. (In German.)

1944

[51] Edler, R., "Der Weg von den Schaltungsbedingungen zum Stromlaufschaltplan" (From the switching operation requirements to the switching circuit diagram), *Elektrotech. u. Maschinenbau*, vol. 62, no. 3–4, pp. 30–52; January, 1944. (In German.)
[52] Shestakov, V. I., "Ob odnom simbolicheskom ischislenii, primenimon k teorii relejnykh elektricheskikh skhem" (On a certain symbolic computation, applicable in the theory of electrical relay circuits), *Uchenye zapiski MGU* (Scientific records of Moscow State University), *Matematika*, vol. 73, kn. 5, pp. 45–48; 1944. (In Russian.)
[53] Stabler, E. R., "Boolean representation theory," *Am. Math. Monthly*, vol. 51, pp. 129–132; 1944.

1945

[54] Gavrilov, M. A., "Metody sinteza relejno-kontaktnykh skhem" (Synthesis methods of relay contact networks), *Elektrichestvo*, no. 2, pp. 54–59; 1945. (In Russian.)
[55] Gavrilov, M. A., "Relejno-kontaktnye skhemy s ventil'nym elementami" (Relay contact networks with rectifying elements), *Izv. AN SSSR*, no. 3, pp. 153–164; 1945. (In Russian.)
[56] Gavrilov, M. A., "Opredelenie chisla kontaktov v skhemakh relejno-kontaktnykh deshifratorov i ikh raspredelenie po rele" (Determination of the number of contacts in relay decoding networks and their distribution between the relays), *Izv. AN SSSR*, no. 12, pp. 1109–1127; 1945. (In Russian.)
[57] Quine, W. V., "On ordered pairs," *J. Symbolic Logic*, vol. 10, pp. 95–96; 1945.

1946

[58] Birkhoff, G. D., and G. Birkhoff, "Distributive postulates for systems like Boolean algebra," *Trans. Am. Math. Soc.*, vol. 60, pp. 3–11; July, 1946.
[59] Oehler, K., "Ein Beitrag zur Kenntnis der Schaltungstechnik" (Contribution to the knowledge of switching circuit technique), *Bull. assoc. suisse électriciens*, vol. 37, no. 11, pp. 298–302; June, 1946. (In German.)
[60] Plechl, O., "Zur Ermittlung elektrischer Kontaktschaltungen" (On investigating electrical contact networks), *Elektrotech. u. Maschinenbau*, vol. 63, no. 1–2, pp. 34–38; January–February, 1946. (In German.)
[61] Plechl, O., and A. Duschek, "Grundzüge einer Algebra der elektrischen Schaltungen" (Principles of an algebra for electric contact networks), *Oesterr. Ingr-Arch.*, vol. 1, no. 3, pp. 203–230; 1946. (In German.)
[62] Ramlau, P., "Matematicheskij analiz skhem avtoblokirovki" (Mathematical methods for analyzing automatic interlocking circuits), *Tekh. Zheleznykh Dorog*, no. 10–11, pp. 20–22; 1946. (In Russian.)
[63] Shestakov, V. I., "Predstavlenie kharakteristicheskikh funktsij predlozhenij, posredstvom vyrazhenij, realizuemykh relejno-kontaktnykh skhemami" (Introduction of the characteristic functions of proposition, by means of expressions realized by relay contact networks), *Izv. AN SSSR*, Ser. mat., no. 10, pp. 529–554; 1946. (In Russian.)

Books:

[64] Packard, C. A., "Relay Engineering," Struthers-Dunn, Inc., Philadelphia, Pa., 640 pp., 44 refs.; 1946.

Dissertations:

[65] Gavrilov, M. A., "Strukturnyj analiz i sintez relejno-kontaktnykh skhem" (Structural analysis and synthesis of relay contact networks), Inst. Avtomat. i Telemekh., AN SSSR, Moscow; 1946. (In Russian.)

1947

[66] Bryleev, A. M., "Teorema obratnogo porjadka i sintez relejno-kontaktnykh skhem" (Theorems of inversions and the synthesis of relay contact networks), *Tekh. Zheleznykh Dorog*, no. 1, pp. 12–15; 1947. (In Russian.)
[67] Duschek, A., "Über eine neue Art von algebraischen Bereichen" (On a new area within the algebraic fields), *Monatsh. Math.*, vol. 52, p. 89; 1947. (In German.)
[68] Gavrilov, M. A., "Ob odnom obshchem metode preobrazovanija relejno-kontaktnykh skhem" (On a certain general method of transformation of relay contact network), *Avtomat. i Telemekh.*, vol. 8, no. 2, pp. 89–107; 1947. (In Russian.)
[69] Gavrilov, M. A., "Strukturnaja klassifikatsija relejno-kontaktnykh skhem" (Structural classification of relay contact networks), *Avtomat. i Telemekh.*, vol. 8, no. 4, pp. 297–307; 1947. (In Russian.)
[70] Gavrilov, M. A., "Analiz relejno-kontaktnykh skhem" (Analysis of relay contact networks), *Elektrichestvo*, no. 4, pp. 5–13; 1947. (In Russian.)
[71] Volotskij, A., "Elementy teorii relejno-kontaktnykh skhem" (Elements of the theory of relay contact networks), *Vestnik svjazi* (*Elektrosviaz'*), no. 7, pp. 16–18, 1947. (In Russian.)

1948

[72] Birkhoff, G., "Lattice theory," *Am. Math. Soc. Colloq. Publ.*, vol. 25, p. 133; 1948.
[73] Bryleev, A. M., "Teoreticheskie metody sinteza relejno-kontaktnykh skhem klassa N" (Theoretical methods for synthesizing relay contact networks of class N), *Tekh. Zheleznykh Dorog*, no. 8, pp. 23–24; 1948. (In Russian.)
[74] Gavrilov, M. A., "Preobrazovanie skhem klassa N" (Transformation of relay contact networks of the class N), *Dokl. AN SSSR*, vol. 59, no. 9, pp. 1579–1582; 1948. (In Russian.) Translation monthly, no. R-3214.
[75] Gavrilov, M. A., "Postroenie relejno-kontaktnykh skhem s mostikovymi soedinenijama" (Design of relay contact networks with bridge connections), *Avtomat. i Telemekh.*, vol. 9, no. 6, pp. 466–479; 1948. (In Russian.)
[76] Gavrilov, M. A., and I. Gluzman, "Primenenie analiticheskikh metodov v postroenii skhem S.Ts.B." (Application of analytical methods in the construction of the circuits of the S.Ts.B.), *Avtomat. i Telemekh.*, vol. 9, no. 1, pp. 74–83; 1948. (In Russian.)
[77] Montgomerie, G. A., "Sketch for an algebra of relay and contactor circuits," *J. IEE*, pt. 2, vol. 95, pp. 355–63, June, 1948; pt. 3, vol. 95, pp. 303–311, July, 1948.
[78] Ramlau, P., "Primenenie algebry logiki dlja analiza skhem i S.Ts.B." (Application of logical algebra in analyzing the circuits of S.Ts.B.), Izd. Leningrad Elektrotekh. in-ta inzhenerov Signalizatsii i Svjazi; 1948.
[79] Shatsev, N. Z., "Elementy teorii relejno-kontaktnykh skhem" (Elements of the theory of relay contact networks), *Trudy Voennotransportoj Akad.*, no. 13, pp. 129–133; 1948. (In Russian.)

1949

[80] Aranovich, B. I., "Ispol'zovanie matrichnykh metodov v voprosakh strukturnogo analiza relejno-kontaktnykh skhem" (Application of matrix methods in problems of the structural analysis of relay contact networks), *Avtomat. i Telemekh.*, vol.

10, no. 6, pp. 437–451; 1949. (In Russian.) *J. Symbolic Logic*, vol. 21, pp. 103–104; March, 1956. (Review by Z. Pawlak.)
[81] Everett, C. J., and G. Whaples, "Representation of sequences of sets," *Am. J. Math.*, vol. 71, pp. 287–293; April, 1949.
[82] Feldbaum, A. A., "The simplest relay system for automatic regulation," *Avtomat. i Telemekh.*, vol. 10, no. 4, pp. 249–266; 1949. (Original in Russian.)
[83] Gavrilov, M. A., "Relejno-kontaktnye deshifratory ispol'zovaniem neskol'kikh impulsnykh priznakov" (Relay contact network decoders employing some impulse technical characteristics), *Avtomat. i Telemekh.*, vol. 10, no. 2, pp. 157–183; 1949. (In Russian.)
[84] Gavrilov, M. A., "K voprosu ob analize relejno-kontaktnykh skhem" (On the question of analyzing relay contact networks), *Dokl. AN SSSR*, vol. 69, no. 2, pp. 181–184; 1949. (In Russian.)
[85] Keister, W., "The logic of relay circuits," *Trans. AIEE*, pt. I, vol. 68, pp. 571–576; 1949.
[86] Ritchie, A. E., "Sequential aspects of relay circuits," *Trans. AIEE*, pt. I, vol. 68, pp. 577–581; 1949.
[87] Shannon, C. E., "The synthesis of two-terminal switching circuits," *Bell Sys. Tech. J.*, vol. 28, pp. 59–98; January, 1940.
[88] Sørvik, R., "Kontakt-ligninger" (Contact equations), *Elektroteк. Tidskr.*, vol. 62, pp. 120–125; April, 1949. (In Norwegian.)
[89] Varnum, E. C., "Relay circuit analysis by odd-even algebra," *Machine Design*, vol. 21, no. 12, pp. 137–139, 192–193; December, 1949.
[90] Washburn, S. H., "Relay 'trees' and symmetric circuits," *Trans. AIEE*, pt. I, vol. 68, pp. 582–586; 1949.

1950

[91] Buffery, G. H., "A contribution to the algebra of relay and switch contacts," *Proc. IEE*, pt. I, vol. 97, pp. 357–362; November, 1950. (Reprint No. 1037.)
[92] Edler, R., "Ein neuer, vielseitig verwendbarer Packetschalter, Entwicklungswege mit Hilfe der Schaltlehre" (New multipurpose packet switch, development directions by means of the switching circuit theory), *Elektrotech. u. Maschinenbau*, vol 67, no. 9, pp. 275–280; 1950. (In German.)
[93] Edler, R., "Entwurf einer Signalanlage nach den Prinzipien der Schaltlehre" (Concepts of a signal system based on the principles of the switching circuit theory), *Ö. T. F.*, vol. 4, no. 7–8, pp. 94–99; 1950. (In German.)
[94] Gavrilov, M. A., and V. A. Khvoshchuk, "Metod chastichnoj inversii v relejnykh skhemakh" (Method for partial inversion in relay circuits), *Dokl. AN SSSR*, vol. 75, no. 5, pp. 685–687; 1950. (In Russian.)
[95] Kalicki, J., "A note on truth tables," *J. Symbolic Logic*, vol. 15, pp. 174–181; 1950.
[96] Kalicki, J., "A test for the existence of tautologies according to many-valued truth tables," *J. Symbolic Logic*, vol. 15, pp. 182–184; 1950.
[97] Lunts, A. G., "Prilozhenie matrichnoj Bulevoj algebry k analizu i sintezu relejno-kontaktnykh skhem" (Application of Boolean matrix algebra to the analysis and synthesis of relay contact networks), *Dokl. AN SSSR*, vol. 70, no. 3, pp. 421–423; 1950. (In Russian.) *J. Symbolic Logic*, vol. 21, p. 104; March, 1956. (Review by Z. Pawlak.)
[98] Lunts, A. G., "Sintez i analiz relejno-kontaktnykh skhem s pomosch'u kharakteristicheskikh funktsij" (Synthesis and analysis of relay contact networks with the aid of characteristic functions), *Dokl. AN SSSR*, vol. 75, no. 2, pp. 201–204; 1950. (In Russian.) *J. Symbolic Logic*, vol. 22, p. 378; December, 1957. (Review.)
[99] Pot, J. J., and G. A. Montgomerie, "Discussion of 'Sketch for an algebra for relay and contactor circuits,' by G. A. Montgomerie," *Proc. IEE*, pt. II, vol. 97, no. 58, pp. 543–544, 1950; pt. III, no. 50, pp. 460–461, 1950; pt. III, vol. 98, no. 51, pp. 75–76, 1951. (*See also* [77].)
[100] Tutugan, F., "O remarcabila aplicare a algebrei logicii in practica construirii schemelor cu contacte de relee" (On special applications of logical algebra in the synthesis of relay contact networks), *Analele Romano-Sovietice*, Ser. mat., fiz., chim., no. 3, pp. 87–92; September, 1950. (In Rumanian.)

Books:

[101] Gavrilov, M. A., "Teorija relejno-kontactnykh skhem" (Theory of relay contact networks), AN SSSR Press, Moscow-Leningrad, 305 pp., 41 bibliog.; 1950. VEB Verlag Technik, Berlin, 324 pp.; 1953. (German translation.)
[102] Markhaj, E. V., and I. A. Babitskij, "Avtomaticheskaja Telefonija" (Automatic telephony), Svjaz'izdat, Moscow; 1950. (In Russian.)

1951

[103] Bellomi, C., "L'algebra dei circuiti elettrici variabili" (Algebra of switching circuits), *Ingegnere*, vol. 25, no. 3, pp. 267–272, March, 1951; no. 4, pp. 377–381, April, 1951; no. 5, pp. 491–496, May, 1951. (In Italian.)
[104] Cherry, E. C., "A history of the theory of information," *Proc. IEE*, pt. III, vol. 98, pp. 383–393, Paper No. 1177; September, 1951.
[105] Duschek, A., "Die Algebra der elektrischen Schaltungen" (Algebra of electrical contact networks), *Rend. mat. applic.*, vol. 10, Rome; 1951. (In German.)
[106] Gilbert, E. N., and M. A. Gavrilov, "Teorija relejno-kontaktnykh skhem," Izdat, AN SSSR, Moscow-Leningrad; 1950. (*See* [101].) *Math. Rev.*, vol. 12, no. 3, p. 225; 1951.
[107] Gilbert, E. N., "N-terminal switching circuits," *Bell Sys. Tech. J.*, vol. 30, no. 3, pp. 668–688; July, 1951.
[108] Lewis, I. A. D., "A symbolic method for the solution of switching and relay circuit problems," *Proc. IEE*, pt. II, vol. 98, pp. 181–191; May, 1951.
[109] Lyndon, R. C., "Identities in two-valued calculi," *Trans. Am. Math. Soc.*, vol. 71, pp. 457–465; 1951. *J. Symbolic Logic*, vol. 18, pp. 69–70; March, 1953. (Review by H. E. Vaughan.)
[110] McCallum, "Mechanized reasoning, logical computors and their design," *Electronic Engrg.*, vol. 23, no. 4, pp. 126–133; April, 1951.
[111] Patterson, G. W., "Logical syntax and transformation rules," *Ann. Computation Lab. Harvard Univ.*, vol. 26, pp. 125–133; 1951.
[112] Piesch, J., "Systematik der automatischen Schaltungen" (The systematism of automatic switching systems), *Ö. T. F.*, vol. 5, no. 3–4, pp. 75–76; 1951. (In German.)
[113] Prager, E., "Theoretické řešeni reléových schemat" (Theoretical solutions of relay contact networks), *Slaboproudý obzor*, vol. 12, no. 8, pp. 183–186; 1951. (In Czech.)
[114] Szukszta, W., "Analiza graficzna schematow teletechnicznych" (Graphical analysis of teletechnical systems), *Przeglad Telekomun.*, vol. 18, (24), no. 7–8, pp. 206–217; 1951. (In Polish.)
[115] Varnun, E. C., "Three-relay circuits," *Machine Design*, vol. 23, no. 2, pp. 121–124, 192, 194–196, 198; 1951.

Books:

[116] Keister, W., A. E. Ritchie, and S. H. Washburn, "The Design of Switching Circuits," D. Van Nostrand Co., Inc., New York, N. Y., 556 pp.; 1951. *J. Symbolic Logic*, vol. 18, pp. 347–348; December, 1953. (Review by A. Church.)

1952

[117] Ashenhurst, R. L., "The application of counting techniques," *Proc. Assoc. Comp. Mach.*, Pittsburgh, Pa., pp. 293–305; May, 1952.
[118] Ashenhurst, R. L., "The Decomposition of Switching Functions," Harvard Comput. Lab., Cambridge, Mass., Rept. No. BL-1, Section II; 1952.
[119] Bellomi, C., "La notazione perfezionata dei contatti nell'algebra dei circuiti elettrici variabili" (Perfect notation for contacts in the algebra of electrical switching circuits), *Ingegnere*, vol. 26, pp. 811–818; July, 1952. (In Italian.)
[120] Berkeley, E. C., "Algebra in electronic design," *Radio-Electronics*, pp. 55–58, February, 1952.
[121] Booth, A. D., "On optimum relation between circuit elements and logical symbols in the design of electronic calculators," *J. Brit. IRE*, vol. 12, no. 12, pp. 587–594; 1952.
[122] Bowman, J. R., "Reduction of the number of possible Boolean functions," *Proc. 9th Macy Conf.*, pp. 120–126; 1952.
[123] Bozzoli, G. R., W. Cormack, F. W. Stutterheim, *et al.*, "Discussion of R. Braae: 'An introduction to Boolean algebra and its applications to electrical relay circuits'," *Trans. S. African IEE*, vol. 43, pt. 9, pp. 251–252; 1952. (See [124].)
[124] Braae, R., "An introduction to Boolean algebra and its applications to electrical relay circuits," *Trans. S. African IEE*, vol. 43, pt. 9, pp. 237–251; 1952.
[125] Burkhart, W. H., "Theorem minimization," *Proc. Assoc. Comput. Mach.*, Pittsburgh, Pa., pp. 259–263; May, 1952.
[126] Burkhart, W. H., "A method for synthesis of two-valued feedback circuits," *Proc. Assoc. Comp. Mach.*, Pittsburgh, Pa., pp. 265–272; May, 1952.
[127] Cardot, C., "Quelques résultats sur l'application de l'algèbre de Boole à synthèse des circuits à relais" (Some results on the application of Boolean algebra to the synthesis of relay circuits), *Ann. télécommun.*, vol. 7, no. 2, pp. 75–84; February, 1952. (In French.)
[128] Gavrilov, M. A., "Vydelenie v relejnykh skhemakh tsepej, vozdejstvujushchykh na dannyj element" (Isolating the cir-

cuits which are activating a given element in a relay network), *Dokl. AN SSSR*, vol. 87, no. 3, pp. 413–416; 1952. (In Russian.)
[129] Gavrilov, M. A., "Opredelenie posledovatel'nosti rabotny elementeviv relejnykh skhemakh" (Determination of the operating sequence of the elements in a relay network), *Avtomat. i Telemekh.*, vol. 13, no. 5, pp. 583–591; 1952. (In Russian.)
[130] Goodell, J. D., "The foundations of computing machinery," *J. Comp. Mach.*, vol. 1, no. 1, pp. 1–13; 1952.
[131] Goodell, J. D., "Decision elements," *Radio and Television News*, Radio-Electronics Engrg. Ed., vol. 48, no. 1, pp. 3–5; no. 2, pp. 14–16, 23; no. 3, pp. 14–16, 31; no. 4, pp. 10–12; 1952.
[132] Jurasov, A. N., "K voprosu sostavlenija strukturnykh formul mnogotaktnykh skhem" (On the question of setting up structural formulae for sequential circuits), *Dev. and Elements of Automation and Remote Control*, Moscow, pp. 125–146; 1952. (In Russian.)
[133] Kalicki, J., "A test for the equality of truth-tables," *J. Symbolic Logic*, vol. 17, pp. 161–163; 1952.
[134] Kalin, T. A., "Formal logic and switching circuits," *Proc. Assoc. Comput. Mach.*, Pittsburgh, Pa., pp. 251–257; May, 1952. *J. Symbolic Logic*, vol. 18, pp. 345–346, December, 1953. (Review.)
[135] Leavitt, M. S., "Boolean algebra and circuit analysis," *Nebraska Blue Print*, vol. 52, no. 5, pp. 13–14, 28; 1952. *J. Symbolic Logic*, vol. 23, no. 1, p. 62; March, 1958. (Review.)
[136] Livovschi, L., "Aplicarea calculului implicatoilor la proiectarea circuitelor cu contacte de relee" (Application of implicator calculus in the design of relay contact networks), *Acad. rep. Populare Romîne*, Bul. stiint., Sect. mat. fiz., vol. 4, no. 1, pp. 195–225, 1952. (In Rumanian.)
[137] Luce, R. D., "A note on Boolean matrix theory," *Proc. Am. Math. Soc.*, vol. 3, pp. 382–388; 1952.
[138] Lunts, A. G., "Algebraicheskie metody analiza i sinteza kontaktnykh skhem" (Algebraic methods of analysis and synthesis of contact networks), *Izv. AN SSSR*, vol. 16, no. 5, pp. 405–426; 1952. (In Russian.) *J. Symbolic Logic*, vol. 22, no. 4, p. 378, December, 1957. (Review.)
[139] Obermann, R. M. M., "Schakelalgebra" (Switching algebra), *PTT-Bedrijf*, vol. 4, no. 4, pp. 125–175; October, 1952. (In Dutch.)
[140] Puig-Adam, P., "Métodos gráfico y algebraico para el proyecto de circuitos electrónicos de cálculo" (Graphical and algebraic methods for the design of electronic computing circuits), *Rev. cienc. apl.*, vol. 6, pp. 289–302, July–August, 1952. (In Spanish.)
[141] Quine, W. V., "The problem of simplifying truth functions," *Am. Math. Monthly*, vol. 59, pp. 521–531; 1952. *J. Symbolic Logic*, vol. 18, pp. 280–282; September, 1953. (Review.)
[142] Schmitz, W., "Schaltungsmethoden" (Methods of connecting switching elements), *Signal u. Draht*, vol. 45, no. 6–7, pp. 91–109; 1952. (In German.)
[143] Schwab, H., "L'algèbre des chaines de contacts" (Algebra of contact networks), *Ann. télécommun.*, vol. 7, no. 1, pp. 2–16; January, 1952. *Rev. gén. elec.*, vol. 61, no. 2, pp. 73–85; February, 1952. (In French.)
[144] Semon, W., "Characteristic numbers and their use in the decomposition of switching functions," *Proc. Assoc. Comput. Mach.*, Pittsburgh, Pa., pp. 273–280; May, 1952.
[145] Staehler, R. E., "An application of Boolean algebra to switching circuit design," *Bell Sys. Tech. J.*, vol. 31, no. 2, pp. 280–305; March, 1952.
[146] Tsetlin, M. L., "Primenenie matrichnogo ischislenija k sintezu relejno-kontaktnykh skhem" (Application of matrix calculations in the synthesis of relay contact networks), *Dokl. AN SSSR*, vol. 86, no. 3, pp. 525–528; 1952. (In Russian.)
[147] Veitch, E. W., "A chart method for simplifying truth functions," *Proc. Assoc. Comput. Mach.*, Pittsburgh, Pa., pp. 127–133; May, 1952.
[148] Vonhot, "Relaisschalttechnik von M. A. Gavrilov, Moskau 1950" ('Relay contact networks,' by M. A. Gavrilov, Moscow, 1950), *Deut. Elektrotech.*, vol. 6, no. 3, p. 140; March, 1952. (In German, *see* [101].)
[149] Wagner, K. W., "Mathematischen Methoden in der Elektrotechnik" (Mathematical methods of electrical engineering), *Elektrotech. Z.*, ETZ, no. 19, pp. 613–619; 1952. (In German.)

Books:

[150] Edler, R., "Schaltlehre, Entwicklung und Anwendung" (Switching theory, development and application), Deuticke, Vienna, 3rd ed., completely rewritten, 208 pp.; 1952.

1953

[151] Ashenhurst, R. L., "A method for synthesizing multiple-output networks sequentially," *Bell Lab. Repts.*, no. 1, pp. V1–V6; 1953.
[152] Ashenhurst, R. L., "A Method for Determining Functional Invariance," Harvard Comput. Lab., Cambridge, Mass., Rept. no. BL-2, Sec. II, pp. 1–12; 1953.
[153] Ashenhurst, R. L., "Non-Disjoint Decomposition," Harvard Comput. Lab., Cambridge, Mass., Rept. no. BL-4, Sec. IV; 1953.
[154] Bellomi, C., "I contatti speciali e quelli dei relé ritardi nell'algebra dei circuiti variabili" (Special contacts and retarded relay contacts in the algebra of switching circuits), *Ingegnere*, vol. 27, no. 3, pp. 287–291; March, 1953. (In Italian.)
[155] Burks, A. W., and J. B. Wright, "Theory of logical nets," Proc. IRE, vol. 41, no. 10, pp. 1357–1365; October, 1953.
[156] Carter, W. C., and A. S. Rettig, "Analytical minimization methods in conjuctive forms," *J. Comput. Sys.*, vol. 1, no. 3, pp. 179–195; July, 1953.
[157] Clos, C., "A study of non-blocking switching networks," *Bell Sys. Tech. J.*, vol. 32, pp. 406–424; 1953.
[158] Copi, I., and F. Harary, "Some properties of N-adic relations," *Portugaliae Mathem.*, vol. 12, pp. 143–152; 1953. *J. Symbolic Logic*, vol. 21, p. 321; September 1956. (Review by K. E. Aubert.)
[159] Craven, T. L., "Logic and the circuit designers," *Electronic Engrg.*, vol. 25, no. 304, pp. 257–259; 1953.
[160] Davis, R. L., "The number of structures of finite relations," *Proc. Am. Math. Soc.*, vol. 4, pp. 486–495; 1953.
[161] Gavrilov, M. A., "Postroenie relejnykh skhem s mostikovymi soedinenijami, iskhodja iz uslovij nesrabatyvanija" (Design of relay networks with bridge connections, starting from the conditions of non-operations), *Avtomat. i Telemekh.*, vol. 14, no. 2, pp. 188–198; 1953. (In Russian.)
[162] Gorgas, J. W., "Sequence charts for switching circuits," *Bell Labs. Record*, vol. 31, no. 12, pp. 492–496; December, 1953.
[163] Karlsson, S. A., "Reläalgebra" (Relay algebra), *Tek. Fören. i Finland Förh.*, vol. 3, no. 3, pp. 45–54; 1953. (In Swedish.)
[164] Karnaugh, M., "The map method for synthesis of combinational logic circuits," *Trans. AIEE (Commun. and Electronics)*, vol. 72, no. 9, pp. 593–598; November, 1953. (See discussion, pp. 598–599.)
[165] Kolpinski, A., "Analiza ukladow stykowo-przekaźnikowych" (Analysis method for contact transmittance), *Przeglad Telekomun.*, vol. 26 (20), no. 12, pp. 380–384; 1953. (In Polish.)
[166] Labutin, A. V., and Ja. G. Belkin, "Retsenzija na knigu M. A. Gavrilova: 'Teorija Relejno-Kontaktnykh Skhem,' " (Review of the book by M. A. Gavrilov: 'Theory of relay contact networks'), *Avtomat. i Telemekh.*, vol. 14, no. 1, pp. 118–119; 1953. (In Russian.) (See [101].)
[167] Leavitt, M. S., "Algebras de Boole e análise de circuitos" (Boolean algebra and circuit analysis), *Gaz. Matem.*, vol. 14, no. 55, pp. 4–7; 1953. (In Portuguese.)
[168] Luce, R. D., "Networks satisfying minimality conditions," *Am. J. Math.*, vol. 75, pp. 825–838; October, 1953.
[169] Moisil, Gr. C. "Lectii asupra teoriei algebrice a mecanismelor automate" (Lectures on the algebraic theory of automatic devices), *Profeste la I.C.E.T.*, Litografiate, Bucuresti; 1953. (In Rumanian.)
[170] Novotný, M., "Theoretické řešení reléových řelězů" (Theoretical solution of relay networks), *Slaboproudý obzor*, vol. 14, no. 7–8, pp. 309–316; July, 1953. (In Czech.)
[171] Oden, H., "Das Relaisdiagram, ein Ausdrucksmittel des Schaltungsingenieurs" (The relay diagram, a method of expression for the switching circuit engineer), *Fernmeldetech. Z.*, vol. 6, pp. 325–327; July, 1953. (In German.)
[172] Pirson, R., "Une théorie de la commutation électrique exposée au moyen de l'algèbre de la logique" (Theory of electric switching networks developed on basis of the algebra of logics), *H. F. Elec. Cour. f., Electronique*, vol. 2, no. 5, pp. 109–116; 1953. (In French.)
[173] Quine, W. V., "Two theorems about truth functions," *Bol. Soc. Math. Mex.*, vol. 10, pp. 64–70; 1953.
[174] Raspanti, M., "A complex algebra for relay circuits," *Elec. Engr.*, vol. 72, no. 11, pp. 992–993; November, 1953.
[175] Riguet, J., "Sur le rapport entre les concepts de machine de multipole et de structure algébrique" (On the status of the principles of multiterminal machines and algebraic structures), *Compt. rend. acad. sci. France*, vol. 237, no. 6, pp. 425–427; 1953. (In French.)
[176] Roginskij, V. N., and A. D. Kharkevich, "O smeshannom gruppoobrazovanii dlja shagovykh ATC" (Miscellaneous group combinations in the stages of ATC), *Vestnik svjazi*, no. 9; 1953. (In Russian.)
[177] Semon, W. L., "Circuit matrices," *Bell Lab. Repts.*, no. 5, pp. VIII 1—VIII 59; 1953.
[178] Serrell, R., "Elements of Boolean algebra for the study of information-handling systems," Proc. IRE, vol. 41, pp. 1366–1380; October, 1953.

[179] Shannon, C. E., and E. F. Moore, "Machine aid for switching circuit design," PROC. IRE, vol. 41, pp. 1348–1351; October, 1953.
[180] Shekel, J., "Sketch for an algebra of switchable networks," PROC. IRE, vol. 41, pp. 913–921; July, 1953.
[181] Shestakov, V. I., "Modelirovanie operatsij ischislenija predlozhenij posredstvom prostejshikh chetyrekhpoljusnykh skhem" (Simulation of the operations of the calculus of propositions by means of four-terminal networks), *Vychisl. Mat. Tekh.*, vol. 1, pp. 56–89; 1953. (In Russian.) IRE TRANS. ON CIRCUIT THEORY, vol. CT-3, p. 79; March, 1956. (Review.)
[182] Singer, T., "The Decomposition Chart as a Theoretical Aid," Harvard Comput. Lab., Cambridge, Mass., Rept. no. BL-4, Sec. III; 1953.
[183] Slepian, D., "On the number of symmetry types of Boolean functions of N variables," *Can. J. Math.*, vol. 5, no. 2, pp. 185–193; 1953. *J. Symbolic Logic*, vol. 20, p. 70; March, 1955. (Review.)
[184] Sobociński, B., "On a universal decision element," *J. Comput. Sys.*, vol. 1, no. 2, pp. 71–80; 1953.
[185] Svoboda, F., "Neurčitá dvouhodnotová Booleova funkce" (The indeterminate two-valued Boolean functions), *Časopis pro Pestování Matem.*, vol. 78, no. 4, pp. 373–375; 1953. (In Czech.)
[186] Touchais, M., "Les applications techniques de la logique" (Technical application of logic), *Mém. soc. ing. civils France*, vol. 106, no. 79; 1953. (In French.)
[187] Washburn, S. H., "An application of Boolean algebra to the design of electronic switching circuits," *Trans. AIEE (Commun. and Electronics)*, vol. 72, no. 8, pp. 380–388; September, 1953.

Books:

[188] Gavrilov, M. A., "Relaisschalttechnik für Stark- und Schwachstromsanlagen" (Relay contact theory for electric and electronic systems), VEB Verlag Technik, Berlin, 324 pp., 41 refs.; 1953. (German translation from the Russian, *see* [101].)
[189] Karmazov, M. G., "Avtomaticheskaja telefonija" (Automatic telephony), Svjaz'izdat Press, Moscow, 3rd ed., 290 pp.; 1953. (In Russian.)

Dissertations:

[190] Adams, R. P., "An Extension of Switching Algebra as an Aid in Synthesizing Logical Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 98 pp.; 1953.
[191] Huffman, D. A., "The Synthesis of Sequential Switching Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 154 pp.; 1953.
[192] McCluskey, E. J., Jr., "A Method for Design of Sequential Switching Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 95 pp.; 1953.
[193] Unger, S. H., "Application of Hypercube Diagrams to the Design of Switching Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 86 pp.; 1953.
[194] Wengert, R. E., "Minimizing Realizations of Sequential Relay Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 93 pp.; 1953.

1954

[195] Ashenhurst, R. L., "The Theory of Abstract Two-Terminal Switching Networks," Harvard Comput. Lab., Rept. no. BL-5, Sec. VII, pp. 1–15; 1954.
[196] Badillo-Barallat, M. C., "El álgebra de la lógica polivalente en la cibernética del cálculo automático aritmético" (Multivalue algebra of logics in the cybernetics of automatic arithmetic computation), *Rev. cálc. autom. ciber*, vol. 2, no. 6, pp. 3–17; 1954. (In Spanish.)
[197] Bellomi, C., "L'adattamento dei circuiti di relé alla disponibilita dei contatti degli organi di comando" (Adaption of relay circuits to the number of contacts available in control devices), *Ingegnere*, vol. 28, no. 5, pp. 491–496, May, 1954; no. 6, pp. 633–640, June, 1954. (In Italian.)
[198] Berkeley, E. C., "The algebra of states and events," *Sci. Monthly*, vol. 78, pp. 232–242; April, 1954. *J. Symbolic Logic*, vol. 20, pp. 286–287; September 1955. (Review.)
[199] Burks, A. W., *et al.*, "An analysis of a logical machine using parenthesis-free notation," *Math. Tables Other Aids to Comp.*, vol. 8, pp. 53–57; 1954. *J. Symbolic Logic*, vol. 20, pp. 70–71; March, 1955. (Review by R. J. Nelson.)
[200] Burks, A. W., R. McNaughton, C. H. Pollmar, *et al.*, "Complete decoding nets; general theory and minimality," *J. Soc. Ind. Appl. Math.*, vol. 2, pp. 201–243; 1954.
[201] Caldwell, S. H., "Recognition and identification of symmetric switching functions," *Trans. AIEE (Commun. and Electronics)*, vol. 73, no. 12, pp. 142–146; May, 1954.
[202] Cherry, E. C., "Generalized concepts of networks," *Proc. Symp. Information Networks*, Polytech. Inst. Brooklyn, Brooklyn, N. Y., vol. 3, pp. 175–184; 1954.
[203] Cotroneo, A., "L'uso della numerazione binaria riflessa nelle catene a rele" (Method of reflective binary numbering by means of relay chains), *Poste e telecomun.*, vol. 7, no. 10, pp. 487–494; October, 1954. (In Italian.)
[204] Craven, T. L., "Some observation on Boolean algebra," *ATE J.* vol. 10, no. 1, pp. 30–37; 1954.
[205] Craven, T. L., "More about circuits and logics," *Electronic Engrg.*, vol. 26, no. 317, pp. 302–305; July, 1954.
[206] Ejler, A. A., "Analiticheskij metod nakhozhdenija naivygodnejshikh konfiguratsij relejno-kontaktnykh skhem" (Analytical method of determining the optimal configuration of a relay contact network), *Sbornik Nauch. Leningrad Elektrotekh. Inst.*, 4th ed.; 1954. (In Russian.)
[207] Gavrilov, M. A., "Osnovnye formuly sinteza relejnykh skhem" (Basic formulae for the synthesis of relay circuits), *Avtomat i Telemekh.*, vol. 15, no. 6, pp. 521–537; 1954. (In Russian.) IRE TRANS. ON CIRCUIT THEORY, vol. CT-3, p. 79; March, 1956. (Review.)
[208] Gilbert, E. N., "Lattice theoretic properties of frontal switching functions," *J. Math. Phys.*, vol. 33, no. 1, pp. 57–67; April, 1954.
[209] Goldammer, R., "Kontaktalgebra, eine kleine Theorie der Rechenschaltungen" (Contact algebra, a simple theory of computing circuits), *Radio-Mentor*, vol. 20, no. 4, pp. 198–204; 1954. (In German.)
[210] Gréa, R. A., and R. Higonnet, "Étude logique de circuits de contacts" (Logical study of switching circuits), *Rev. gén. élec.*, vol. 63, no. 1, pp. 19–34; January, 1954. (In French.)
[211] Huffman, D. A., "Information conservation and sequence transducers," *Proc. Symp. Information Networks*, Polytech. Inst. Brooklyn, Brooklyn, N. Y., vol. 3, pp. 291–307; 1954.
[212] Huffman, D. A., "Synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, no. 3, pp. 161–190, March, 1954; no. 4, pp. 275–303, April, 1954. *J. Symbolic Logic*, vol. 20, pp. 69–70; March, 1955. (Review by R. J. Nelson.)
[213] Jablonskij, S. V., "Realizatsija linejnoj funktsii v klasse P-skhem" (Realization of a linear function in the class of series-parallel networks), *Dokl. AN SSSR*, vol. 94, no. 5, pp. 805–806; 1954. (In Russian.) IRE TRANS. ON CIRCUIT THEORY, vol. CT-3, p. 78; March, 1956. (Review.)
[214] Kalicki, J., "An undecidable problem in the algebra of truth tables," *J. Symbolic Logic*, vol. 19, pp. 172–176; 1954.
[215] Karnaugh, M., "Map method for the synthesis of logical circuits," *Elec. Engrg.*, vol. 73, no. 2, p. 136; 1954.
[216] Lee, C. Y., "Switching functions on an N-dimensional cube," *Trans. AIEE (Commun. and Electronics)*, vol. 73, no. 14, pp. 289–291; September, 1954.
[217] Markov, A. A., "O nepreryvnosti konstruktivnykh funktsij" (On the continuity of constructive functions), *Uspekhi Mat. Nauk*, vol. 9, no. 3 (61), pp. 226–230; 1954. (In Russian.) *J. Symbolic Logic*, vol. 21, pp. 319–320, September, 1956. (Review by A. Ehrenfeucht.)
[218] Moisil, Gr. C., "Algebra schemelor cu elemente ventil" (Algebra of switching circuits with rectifying elements), *Rev. Univ. C. I. Parhon si Politech.*, Bucuresti, no. 4–5, pp. 9–42; 1954. (In Rumanian.)
[219] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. I. Asupra schemelor cu elemente ventil" (Application of Galoisian fields in the theory of automatic devices. I. In circuits with rectifying elements), *Comun. Acad. rep. populare Romîne*, vol. 4, no. 11–12, pp. 581–585; 1954. (In Rumanian.)
[220] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. II. Scheme cu doua elemente intermediare" (Application of Galoisian fields in the theory of automatic devices. II. Circuits with two intermediate elements), *Comun. Acad. rep. populare Romîne*, vol. 4, no. 11–12, pp. 587–589; 1954. (In Rumanian.)
[221] Muller, D. E., "Application of Boolean algebra to switching circuits design and to error detection," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-3, pp. 6–12; September, 1954.
[222] Namian, P., "Sur l'étude et la réalisation des circuits à coincidence" (On the analysis and synthesis of coincidence circuits), *Onde élec.*, vol. 34 (323), pp. 123–129; 1954. (In French.)
[223] Oberman, R. M. M., "De bewerktekens in de schakelalgebra" (Algebraic symbols in the switching circuit algebra), *PTT-Bedrijf*, vol. 6, no. 4, pp. 1–10; April, 1954. (In Dutch.)
[224] Obermann, R. M. M., "Het ontwerpen van schakelingen. I. De grondslagen van de schakelalgebra" (Principles of the switching circuit algebra), *Ingenieur*, vol. 66, no. 42, pp. E101–E111; October, 1954. (In Dutch.)
[225] Obermann, R. M. M., (The design of circuits), *Ingenieur*, vol. 66, no. 50, pp. E125–E134; December, 1954. (Original in Dutch.)

[226] Okada, S., "Topology applied to switching circuits," *Proc. Symp. Information Networks*, Polytech. Inst. Brooklyn, Brooklyn, N. Y., pp. 267–290; 1954. *J. Symbolic Logic*, vol. 21, pp. 210–211; June, 1956. (Review.)
[227] Parry, W. T., "A new symbolism for the propositional calculus," *J. Symbolic Logic*, vol. 19, pp. 161–168; 1954.
[228] Parry, W. T., "Symbolic methods in the design of delay- and cyclefree logical nets," 1954 IRE National Convention Record, pt. 4, pp. 58–64.
[229] Pearce, J. G., "Timing charts for circuit design," *ATE J.*, vol. 10, pp. 47–53; January, 1954.
[230] Petrucelly, V., "Boolean algebra: New tool for circuit designers," *Elec. Mf.*, vol. 54, no. 2, pp. 97–101; August, 1954.
[231] Povarov, G. N., "O funktsional'noj razdelimosti Bulevykh funktsij" (Functional separability of Boolean functions), *Dokl. AN SSSR*, vol. 94, no. 5, pp. 801–803; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1956. (Review.)
[232] Povarov, G. N., "O sinteze Kontaktnykh mnogopoljusnikov" (Synthesis of multiterminal contact networks), *Dokl. AN SSSR*, vol. 96, no. 6, pp. 1075–1078; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1956. (Review.)
[233] Povarov, G. N., "Matrichnye metody analiza relejno-kontaktnykh skhem po uslovijam nesrabatyvanija" (Matrix methods for analyzing switching circuits from their nonoperating conditions), *Avtomat. i Telemekh.*, vol. 15, no. 4, pp. 332–335; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 79; March, 1956. (Review.)
[234] Righi, R., "L'algebra dei circuiti di commutazione" (Algebra of switching circuits), *Ing. ferroviaria*, vol. 9, no. 1, pp. 33–46, January, 1954. (In Italian.)
[235] Righi, R., "Ulteriori sciluppi dell'algebra di commutazione: applicazione ai circuiti multi-terminali" (Further developments in switching algebra: application to multi-terminal networks), *Ing. ferroviaria*, vol. 9, no. 4, pp. 271–276, April, 1954; no. 5, pp. 397–408, May, 1954. (In Italian.)
[236] Righi, R., "I circuiti sequenziali e la loro analisi" (Sequential circuits and their analysis), *Ing. ferroviaria*, vol. 9, no. 11, pp. 847–856; November, 1954. (In Italian.)
[237] Roginskij, V. N., "K voprosy o sinteze schetnykh skhem" (On the questions of the synthesis of computing circuits), *Sbornik Nauch. raboty provodnoj svjazi*, Izd. AN SSSR; 1954. (In Russian.)
[238] Roginskij, V. N., "Uchet neispol'zuemykh sostojanij pri sinteze relejnokontaktnykh skhem" (Taking unused states into account in the synthesis of relay contact networks), *Avtomat. i Telemekh.* vol. 15, no. 3, pp. 202–222; 1954. (In Russian.)
[239] Rohleder, H., "Der dreiwertige Aussagenkalkül der theoretischen Logik und seine Anwendung zur Beschreibung von Schaltungen, die aus Elementen mit zwei stabilen Zustände bestehen" (Three-valued Boolean algebra and its application in the description of switching circuits consisting of bistable contact elements), *Z. AMM*, vol. 34, no. 819, pp. 308–311; 1954. (In German.)
[240] Samson, E. W., and B. E. Mills, "Circuit Minimization, Algebra and Algorithms for new Boolean canonical expressions," AF Cambridge Research Center, Bedford, Mass., Tech. Rept. No. 54–21; April, 1954.
[241] Schliebs, G., "Über die Grundlagen der Algebra der Logik und ihre Anwendung in der Schaltungstheorie" (Principles of the algebra of logics and its application in switching circuit theory), *Funk u. Ton*, vol. 8, no. 2, pp. 57–71; February, 1954. (In German.)
[242] Shestakov, V. I., "Algebraicheskij metod sinteza avtonomykh sistem dvukpositsionnykh rele" (Algebraic method of synthesis of autonomic systems consisting of two-position relays), *Avtomat. i Telemekh.*, vol. 15, no. 4, pp. 310–324; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1954. (Review.)
[243] Shestakov, V. I., "O preobrazovanii mnogotsiklicheskoj posledovatel'nosti v vozvratnuju" (Transformation of a monocyclic into a recurrent sequence), *Dokl. AN SSSR*, vol. 98, no. 4, pp. 541–544; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1956. (Review.)
[244] Shestakov, V. I., "Algebraicheskij metod sinteza mnogotaktnykh relejnykh sistem" (Algebraic method of synthesis of poly-cyclic relay systems). *Dokl. AN SSSR*, vol. 99, no. 6, pp. 987–990; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1956. (Review.)
[245] Shestakov, V. I., "Algebraicheskij metod analiza avtonomykh sistem dvukpozitsionnykh rele" (Algebraic method of analysis of autonomic systems consisting of two-position relays), *Avtomat. i Telemekh.*, vol. 15, no. 2, pp. 107–123; 1954. (In Russian.) IRE Trans. on Circuit Theory, vol. CT-3, p. 78; March, 1956. (Review.) *J. Symbolic Logic*, vol. 23, no. 2, pp. 231–233; June, 1958. (Review.)
[246] Svoboda, A., "Synthesa reléových sítí" (Synthesis of relay networks), *Stroje na Zpracování Informací*, Prague; pp. 157–208, 1954. (In Czech.)
[247] Tellegen, B. D. H., (Electric circuit theory), *Ingenieur*, vol. 66, no. 42, pp. E101–E111; October, 1954. (Original in Dutch.)
[248] Trent, H. M., "A note on the enumeration and listing of all possible trees in a connected linear graph," *Proc. Natl. Acad. Sci. U. S.*, vol. 40, pp. 1004–1007; October, 1954.
[249] Vogel-Jørgensen, U., "Rele-logik" (Relay logics), *Ingeniøren*, vol. 63, pp. 430–432; May, 1954. (In Danish.)
[250] Washburn, S. H., "Boolean algebra in electronic circuit design," *Elec. Engrg.*, vol. 73, no. 2, p. 164; 1954.
[251] Washburn, S. H., M. Karnaugh, and S. H. Caldwell, "Discussion of S. H. Caldwell: 'The recognition and identification of symmetric switching functions'," *Trans. AIEE (Commun. and Electronics)*, vol. 73, no. 12, pp. 146–147; May, 1954. (See [201].)
[252] Watanabe, T., "Input routing charts for relay networks and their applications," *J. Inst. Elec. Commun. Engrs. Japan*, vol. 37, no. 10, pp. 709–714; October, 1954. (Original in Japanese.)
[253] Zaheb, D., and W. P. Caywood, "A symbolic method for synthesis of two-terminal switching circuits," *Trans. AIEE (Commun. and Electronics)*, vol. 73, no. 16, pp. 690–693; January, 1954.
[254] Zühlsdorf, W., "Methoden der theoretisch-matematischen Behandlung von Aufgaben der Schaltungstechnik" (Theoretical-mathematical methods for solving the problems in the switching circuit technique), *Deut. Elektrotech.*, vol. 8, no. 2, pp. 54–60; 1954. (In German.)
[255] Zühlsdorf, W., "Ein Sowjetisches Buch über Relaisschalttechnik für Starkstroms-und Schwachstromsanlagen" (Soviet-Russian book of relay contact networks for electrical and electronic installations), *Deut. Elektrotech.*, vol. 8, no. 2, pp. 53–54; February, 1954. (In German.) (See [101] and [188].)

Books:

[256] Berkeley, E. C., "A summary of symbolic logic and its practical applications," E. C. Berkeley and Assoc., New York, N. Y.; June, 1954. *J. Symbolic Logic*, vol. 18, p. 68; March, 1953. (Review.)
[257] Markov, A. A., (Theory of Algorithms), Izd. AN SSSR, Moscow-Leningrad; 1954. (Original in Russian.) *J. Symbolic Logic*, vol. 23, pp. 77–79; March, 1958. (Review by G. F. Rose.)
[258] Shestakov, V. I., "Sintez electronnykh vychislitel'nykh i upravljajushchikh skhem" (Synthesis of electronic computing and controlling circuits), Izd. inostrannoj literatury, Moscow; 1954. (In Russian.)

Dissertations:

[259] Lett, A. S., "The Design of Hazard-Free Relay Contact Networks," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 52 pp.; 1954.
[260] Muroga, K., "The Electronic Realization of Symmetric Switching Functions," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 106 pp.; 1954.
[261] Povarov, G. N., "Issledovanie kontaktnykh skhem s minimal'nym chislom kontaktor" (Investigation of contact networks with the minimal number of contacts), Inst. Avtomat. i Telemekh., AN SSSR; 1954. (In Russian.)

1955

[262] Ashenhurst, R. L., "A Uniqueness Theorem for Abstract Two-Terminal Switching Networks," Harvard Comput. Lab., Cambridge, Mass., Rept. No. BL-10, Sec. VI, pp. 1–8; 1955.
[263] Bellomi, C., "Gli sviluppi dell'unità nella semplificazione dei circuiti di relè" (Development of unity in the simplification of relay circuits), *Ing. ferroviaria*, vol. 10, no. 9, pp. 631–641; September, 1955. (In Italian.)
[264] Bennett, W. S., "Minimizing and mapping sequential circuits," *Trans. AIEE (Commun. and Electronics)*, vol. 74, no. 20, pp. 443–447; September, 1955.
[265] Burks, A. W., R. McNaughton, C. H. Pollmar, *et al.*, "The folded tree," *J. Franklin Inst.*, vol. 260, no. 1, pp. 9–24, July; no. 2, pp. 115–126, August, 1955.
[266] Cohen, J. W., "Some examples of the use of implication in switching algebra," *Commun. News*, vol. 16, no. 1, pp. 2–10; October, 1955.
[267] Durst, L. K., "On certain subsets of finite Boolean algebra," *Proc. Am. Math. Soc.*, vol. 6, pp. 695–697; 1955.
[268] Eguchi, S., (A new method of relay circuit symbolization), *J. Inst. Elec. Commun. Engrs. Japan*, vol. 38, no. 11, pp. 898–904; November, 1955. (Original in Japanese.)

[269] Gavrilov, M. A., "Relejnye skhemy s ventil'nymi setkami" (Relay networks with blocking nets), *Avtomat. i Telemekh.*, vol. 16, no. 4, pp. 328–343; 1955. (In Russian.)
[270] Hohn, F. E., "Some mathematical aspects of switching," *Am. Math. Monthly*, vol. 62, pp. 75–90; 1955. *J. Symbolic Logic*, vol. 21, pp. 209–210; June, 1956. (Review.)
[271] Hohn, F. E., "A matrix method for the design of relay circuits," IRE Trans. on Circuit Theory, vol. CT-2, pp. 154–161; June, 1955.
[272] Hohn, F. E., and L. R. Schissler, "Boolean matrices and the design of combinational relay switching circuits," *Bell Sys. Tech. J.*, vol. 34, no. 1, pp. 177–202; January, 1955. *J. Symbolic Logic*, vol. 20, pp. 104–105; March, 1956. (Review by Z. Pawlak.)
[273] Huffman, D. A., "A Study of the Memory Requirements of Sequential Switching Circuits," Electronics Research Lab., Mass. Inst. Tech., Cambridge, Tech. Rept, No. 293; April, 1955.
[274] Ioanin, G., "Asupra teoriei algebrice a contactelor multipozitionale" (On the algebraic theory of multipositional switches), *Acad. rep. populare Romîne*, Bul. stiint., Sec. mat. fiz., vol. 7, no. 2, April–June; 1955. (In Rumanian.)
[275] Itoh, M., " 'n-ti kansusoku' [n-ti ronri] ni tuite" (Networks of n-symbol function [of n-symbol logic],) Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 2, pp. 96–98, 99–101; 1955. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 100; 1957. (Review.)
[276] Ivanov, V. I., "Tsiklicheskie relejny skhemy i analiticheskie sootnoshenija v nikh" (Cyclic relay networks and their analytical relations), *Dokl. AN SSSR*, vol. 104, no. 2, pp. 239–241; 1955. (In Russian.) *J. Symbolic Logic*, vol. 21, p. 333; September, 1956. (Review.)
[277] Klein, M. L., "Simplified method for the design of logical conversion matrices," *Electronic Engrg.*, vol. 27, pp. 270–272; June, 1955.
[278] Kurihara, T., "Yugen tatironri no denki kairo ni yoru hyogen ni tuite" (On the representation of finitely many-valued logics by electric circuits), Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 2, pp. 102–106; 1955. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 100; March, 1957. (Review.)
[279] Lee, C. Y., "Analysis of switching networks," *Bell Sys. Tech. J.*, vol. 34, no. 6, pp. 1287–1315; November, 1955.
[280] Mealy, G. H., "A method for synthesizing sequential circuits," *Bell Sys. Tech. J.*, vol. 34, no. 5, pp. 1045–1079; September, 1955. *J. Symbolic Logic*, vol. 22, pp. 334–335; September, 1957. (Review.) Bell Telephone Sys. Monograph No. 2458.
[281] Mekler, Ja.I., "Sintez skhemy glavnogo privoda prokatnogo stana" (Synthesis of the principal circuits for a rolling-mill power system), *Avtomat. i Telemekh.*, vol. 16, no. 1, pp. 71–86; 1955. (In Russian.)
[282] Moisil, G. C., "O metodă pentru sinteza schemelorcu programe de lucru date" (Method for the synthesis of circuits with time program), *Acad. rep. populare Romîne, Filiala Iaşi, Studii cercetări stiint.*, vol. 6, no. 1–2; 1955. (In Rumanian.)
[283] Moisil, G. C., "Contributii la teoria algebrica a mecanismelor automate" (Contribution to the algebraic theory of automatic devices), *Acad. rep. populare Romîne*, Bul. stiint., Sect. mat. fiz., vol. 7, no. 2; April–June, 1955. (In Rumanian.)
[284] Moisil, Gr.C., "Simplificarea schemelor prin introducerea contactelor multipozitionale" (Circuit simplification by introduction of multipositional contact elements), *Acad. rep. populare Romîne*, Bul. stiint., no. 4; 1955. (In Rumanian.)
[285] Muller, R. K., "On the Synthesis of a Minimal Representation of a Logic Function," AF Cambridge Research Center, Bedford, Mass., Tech. Rept. No. 55–104; April, 1955.
[286] Nelson, R. J., "Simplest normal truth functions," *J. Symbolic Logic*, vol. 20, pp. 105–108; June, 1955. *J. Symbolic Logic*, vol. 21, pp. 328–330; September, 1956. (Review by R. McNaughton.)
[287] Nelson, R. J., "Weak simplest normal truth functions," *J. Symbolic Logic*, vol. 20, pp. 232–234; September, 1955. *J. Symbolic Logic*, vol. 21, pp. 330–331; September, 1956. (Review by R. McNaughton.)
[288] Ninimiya, I., "On the number of types of symmetric Boolean output matrices," *Mem. Fac. Engrg., Nagoya Univ.*, vol. 7, no. 2, pp. 115–124; 1955.
[289] Okada, S., "Algebraic and topological foundations of network synthesis," *Proc. Symp. Network Synthesis*, Polytech. Inst. Brooklyn, Brooklyn, N. Y.; April, 1955.
[290] Ostianu, V. M., "O sinteze kontaktnykh skhem s shagovymi perekljuchateljami" (On the synthesis of switching circuits with selector switches), *Dokl. AN SSSR*, vol. 103, no. 4, pp. 827–830; 1955. (In Russian.)
[291] Parker, W. L., and B. A. Bernstein, "On uniquely solvable Boolean equations," *Univ. Calif. Publ. Math.*, vol. 3, no. 1, pp. 1–29; 1955.
[292] Pelz, F. M., "Elektrische Netzwerke mit Schaltkontakten. Algebra für Kontaktschaltungen" (Electrical networks with switches. Algebra for contact networks), *Fernmeldetech. Z.*, vol. 8, no. 6, pp. 328–334; June, 1955. (In German.)
[293] Piesch, J., "Die Matrix in der Schaltungsalgebra zur Planung Relaisgesteuerter Netzwerke" (Matrices in the switching-circuit algebra for designing relay-controlled networks), *Arch. Elektrichen. Übertr.*, vol. 9, no. 10, pp. 460–468; October, 1955. (In German.)
[294] Postley, J. A., "A method for the evaluation of a system of Boolean algebraic equations," *Math. Tables Other Aids Comp.*, vol. 9, pp. 5–8; 1955.
[295] Povarov, G. N., "O metodike analiza simmetricheskikh kontaktnykh skhem" (Method of analyzing symmetrical contact networks), *Avtomat i Telemekh.*, vol. 16, no. 4, pp. 364–366; 1955. (In Russian.)
[296] Povarov, G. N., "Matematicheskaja teorija sinteza kontaktnykh (1,k)-poljusnikov" (Mathematical theory for the synthesis of (1,k)-terminal contact networks), *Dokl. AN SSSR*, vol. 100, no. 5, pp. 909–912; 1955. (In Russian.) *J. Symbolic Logic*, vol. 21, p. 332; September, 1956. (Review.) IRE Trans. on Circuit Theory, vol. CT-3, p. 79; March, 1956. (Review).
[297] Povarov, G. N., "K izucheniju simmetricheskikh Bulevykh funktsij s tochki zrenija relejno-kontaktnykh skhem" (Concerning the study of symmetric Boolean functions from the point of view of the theory of relay contact networks), *Dokl. AN SSSR*, vol. 104, no. 2, pp. 183–185; 1955. (In Russian.) *J. Symbolic Logic*, vol. 22, p. 99; March, 1957. (Review.)
[298] Povarov, G. N., "Novij metod sintezu simmetrichnikh kontaktnikh skhem" (New method of synthesis of symmetric contact networks), *Dopovidi AN Ukr. RSR*, no. 2, pp. 115–177; 1955. (In Russian.)
[299] Quine, W. V., "A way to simplify truth functions," *Am. Math. Monthly*, vol. 62, pp. 627–631; 1955. *J. Symbolic Logic*, vol. 21, pp. 328–330; September, 1956. (Review by R. McNaughton.)
[300] Righi, R., (The commutation function in general and symmetrical commutation functions in particular), *Ing. ferroviaria*, vol. 10, no. 10, pp. 713–737; October, 1955. (Original in Italian.)
[301] Roginskij, V. N., "Sintez mnogotaktnykh relejnykh skhem" (Synthesis of multicycle relay circuits), *Sbornik Nauch. Rabot Provodnoj Svjazi, AN SSSR*, no. 4, p. 111; 1955. (In Russian.)
[302] Rohleder, H., "Die Verwendung von Aussagenkalkülen zur Beschreibung elektrischer Schaltungen" (Application of symbolic logic in the description of electrical contact networks), *Z. math. Logik Gr. Math.*, vol. 1, pp. 304–309; 1955. (In German.)
[303] Samson, E. W., and R. K. Muller, "Circuit Minimization; Sum to One Process for Irredundant Sums," AF Cambridge Research Center, Bedford, Mass., Tech. Rept. no. 55–118; August, 1955.
[304] Samson, E. W., and R. K. Muller, "Circuit Minimization, Minimal and Irredundant Boolean Sums by Alternative Set Method," AF Research Center, Bedford, Mass., Tech. Rept. No. 55–109; June, 1955.
[305] Schaefer, D. H., "A rectifier algebra," *Trans. AIEE, (Commun. and Electronics)*, vol. 74, pp. 679–682; 1955. *J. Symbolic Logic*, vol. 21, p. 401; December, 1956. (Review by R. J. Nelson.)
[306] Trakhtenbrot, B. A., "Sintez bespovtornykh skhem" (Synthesis of nonrepetitive circuits), *Dokl. AN SSSR*, vol. 103, pp. 973–976; 1955. (In Russian.) *J. Symbolic Logic*, vol. 21, pp. 332–333; September, 1956. (Review.)
[307] van der Poel, W. L., "Einige bijzondere onderwerpen uit de schattelalgebra" (Some characteristic properties of the switching-circuit algebra), *Ingenieur*, vol. 67, no. 1, pp. E9-E14; 1955. (In Dutch.)
[308] Weitzsch, F., "Traditionelle Aussagelogik und Elektronische Rechen- und Schaltanlagen" (Traditional symbolic logic, electronic computing, and switching circuits), *Elektron. Rundschau*, vol. 10, no. 12, pp. 331–334; December, 1955. (In German.)
[309] Wimpey, J. L., "Switching Systems," Symp. Electro-Magnetic Relays, Oklahoma Inst. Tech., Stillwater; March, 1955.
[310] Yasuura, K., "Keidenkikairo ni yoru tatimeidaironri no hyogen ni tuite" (On the representation of many-valued propositional logics by relay circuits), Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 2, pp. 94–96; 1955. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 102; March, 1957. (Review.)
[311] Zemanek, H., "Schaltalgebra" (Switching-circuit algebra), *Radio-Technik*, vol. 31, p. 201; 1955. (In German.)

Books:

[312] Berkeley, E. C., "Circuit Algebra—Introduction," E. C. Berkeley and Assoc., New York, N. Y., revised ed.; August, 1955.
[313] Higonnet, R. A., and R. A. Gréa, "Étude logique des circuits électriques et des systèmes binaires" (Logical study of elec-

trical circuits and binary systems), Berger-Levrault, Paris, 425 pp., 28 bibliog.; 1955. (In French.)
[314] Roginskij, V. N., and A. D. Kharkevich, "Relejnye skhemy v telefonii" (Relay circuits in the telephony), Tipografija Svjazizdata, Moscow, 166 pp., 43 bibliog.; 1955. (In Russian.)

Dissertations:

[315] Burke, T. E., "Electronic Switching Circuits Synthesized from Relay Contact Networks," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 69 pp.; 1955.
[316] Bushnell, J. C., "The Design of Hazard-Free Switching Networks," Gen. Sci. Dept., Mass. Inst. Tech., Cambridge, 20 pp.; 1955.
[317] Hoy, E. C., "Simplifying Switching Functions Using a General Purpose Digital Computer," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 78 pp.; 1955.
[318] Huffman, D. A., "Philosophies of Sequential Circuit Behavior," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 24 pp.; 1955.
[319] Metze, G. A., "Many-Valued Logic and the Design of Switching Circuits," University of Illinois, Urbana; 1955.

1956

[320] Ashenhurst, R. L., "The Structure of Multiple-Coincidence Selection Systems," Harvard Comput. Lab., Cambridge, Mass. Rept. No. BL-14, p. 20; May, 1956. Dissertation, Harvard University, Cambridge; May, 1956.
[321] Barnes, L. A., "Boolean algebra for switching circuits," *Elec. Mf.*, vol. 58, no. 2, pp. 126–127; August, 1956.
[322] Bellomi, C., "L'algebra delle dirette unioni verticali e dei circuiti a ponte equivalenti" (Algebra of direct vertical junctions and their equivalent bridge circuits), *Ingegnere*, vol. 30, no. 3, pp. 249–264; March, 1956. (In Italian.)
[323] Bellomi, C., "I ponti e le dirette unioni verticali complimentari nell'algebra dei circuiti di rele" (Direct vertical combination of bridges in the algebra of relay circuits), *Ing. ferroviaria*, vol. 11, no. 4, pp. 297–316; April, 1956. (In Italian.)
[324] Bing, K., "On simplifying truth-functional formulas," *J. Symbolic Logic*, vol. 21, pp. 253–254; September, 1956.
[325] Blokh, A.Sh., "Sintez kontaktnykh [p,q]-poljusnikov" (Synthesis of [p,q]-terminal contact networks), *Dokl. AN SSSR*, vol. 111, no. 5, pp. 1017–1019; 1956. (In Russian.) *J. Symbolic Logic*, vol. 22, no. 3, pp. 333–334; September, 1956. (Review.)
[326] Constantinescu, P., "Asupra reducerii numarului de contacte prin introducerea circuitelor in punte" (On reduction of the number of contacts by using bridge circuits), *Analele Univ. C. I. Parhon, Bucuresti*, no. 11, pp. 45–67; 1956. (In Rumanian).
[327] Constantinescu, P., "Asupra unor scheme obtinute folosind congruente de numere integri in teoria mecanismelor automate" (On some circuits obtained by the application of integer comparison theory to the theory of automatic devices), *Analele Univ. C. I. Parhon, Bucuresti*, no. 12, p. 23; 1956. (In Rumanian.)
[328] Dragos, V., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. VI. Clasificarea evolutiilor schemelor cu doua elemente intermediare" (Application of Galoisian fields in the theory of automatic devices. VI. Classification of the development of schemes with two intermediate elements), *Bul. Acad. rep. populare Romîne*, vol. 8, no. 1, p. 21; 1956. (In Rumanian.)
[329] Gavrilov, M. A., "Sovremennoe sostojanie teorii relejnykh skhem" (Present state of the theory of relay circuits), in "Telemekh, v. narodnom khozjajstve," AN SSSR, Moscow, pp. 99–133; 1956. (In Russian.)
[330] Goilav, E., "Contributii la tratarea algebrica a schemelor electrice cu relee si contacte multipozitionale" (Application of contact switching theory to circuits with multipositional elements), *Electrotechnica*, no. 3, pp. 130–138; 1956. (In Rumanian.)
[331] Greniewski, M., "Intrebuintarea logicilor trivalente in teoria mecanismelor automate. I. Realizarea priu circuftea funcutiilor fundamentale" (Application of three-valued symbolic logic in the theory of automatic devices. I. Realization of fundamental functions in the circuit design), *Commun. Acad. rep. populare Romîne*, vol. 6, no. 2, pp. 225–229; 1956. (In Rumanian.)
[332] Halmos, P. R., "The basic concepts of algebraic logic," *Am. Math. Monthly*, vol. 63, no. 6, pp. 363–387; June–July, 1956.
[333] Ioanin, G., "Sinteza schemelor in care intra selectorij" (Synthesis of selector switching circuits), *Bul. Acad. rep. populare Romîne, Bucuresti*, vol. 7, no. 3, p. 489; August–September, 1956. (In Rumanian.)
[334] Ioanin, G., and G. C. Moisil, "La synthèse des schèmas à contacts et relais avec des conditions de travail données pour les éléments executifs" (Synthesis of contact relay circuits based on the operating conditions for the active elements), *Acad. rep. populare Romîne, Bucuresti*, vol. 1, no. 2, p. 167; 1956. (In French.)
[335] Itoh, M., "Itegen n-ti kansusoku (ronri) hoteisiki no ippankai ni tuite" (On the general solution of logic equations of the three-symbol functions), Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 4, pp. 239–243; 1956. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 101; 1957. (Review.)
[336] Itoh, M., "Tagen Boole [nitironri] hoteisiki no ippankai ni tuite" (On the general solution of Boolean [two-valued] logical equations with several variables), Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 4, pp. 246–248; 1956. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 101; March, 1957. (Review.)
[337] Itoh, M., "Santironri hoteisiki no ippankai ni tuite" (On the general solution of three-valued logical equations), Kyushu University, Fukuoka, Tech. Rept., vol. 28, no. 4, pp. 248–250; 1956. (In Japanese.) *J. Symbolic Logic*, vol. 22, no. 1, p. 101; March, 1957. (Review.)
[338] Ivanov, V. I., "Issledovanie skhem relejnykh perekljuchatelej" (Investigation of relay contact circuits), in "Telemekh. v narodnom khozjajstve," AN SSSR, pp. 146–158; Moscow, 1956. (In Russian.)
[339] Ivanov, V. I., "Sintez tsiklicheskikh relejno-kontaktnykh skhem s odnotipnoj strukturoj" (Synthesis of cyclic relay contact networks using a single type of structure), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, Moscow, p. 50; 1957. Dissertation, Inst. Avtomat. i Telemekh., 1956. (In Russian.)
[340] Jablonskij, S. V., "Funktsionalnie postroenija v mnogoznachnykh logikakh [Rezjume]" (Functional constructions in multi-valued logics [Summary]), *3rd All-Union Conf. Math.*, AN SSSR, Moscow, pp. 71–73; 1956. (In Russian.)
[341] Jablonskij, S. V., "Ob odnom semejstve klassov funktsij algebry logiki, dopuskajushchikh prostuju skhemnuju realizatsiju" (On a certain set of classes of function in the algebra of logic which allows simple circuit realizations [Summary]), *3rd All-Union Conf. Math.*, AN SSSR, Moscow, vol. 2, p. 149; 1956. (In Russian.)
[342] Kharkevich, A. D., "O postroenii polnodostupnykh kommutatsionnykh skhem" (On the design of fully accessible switching networks), *Sbornik nauch. rabot provodnoj svjazi*, AN SSSR, no. 5, pp. 79–92; 1956. (In Russian.)
[343] Korobkov, V. K., "Realizatsija simmetricheskikh funktsij v klasse *p*-skhem" (Realization of symmetrical functions in a class of *p* functions), *Dokl. AN SSSR*, vol. 109, no. 2, pp. 260–263; 1956. (In Russian.)
[344] Lanning, W. C., "Boolean algebra," *Sperry Engrg. Rev.*, vol. 9, no. 3–4, pp. 8–13, May–June; pp. 14–17, July–August, 1956.
[345] Lazarev, V. G., "Opredelenie minimal'nogo chisla promezhutochnykh rele pri sinteze mnogotaktnykh skhem" (Determination of the minimum number of intermediate relays in the synthesis of multiple cycle networks), *Sbornik nauch. rabot provodnoj svjazi*, AN SSSR, Moscow, no. 5, pp. 93–103; 1956. (In Russian.)
[346] Lee, C. Y., and W. H. Chen, "Several-valued combinational switching circuits," *Trans. AIEE, (Commun. and Electronics)*, vol. 75, no. 25, pp. 278–283; July, 1956. Bell Lab. Monograph no. 2746.
[347] Linsman, M., "Sur une méthode de simplification de circuits dans la théorie des fonctions de commutation" (On a method of circuit simplification in the theory of switching circuits), *Bull. soc. roy. sci. Liège*, vol. 25, pp. 163–166; 1956. (In French.)
[348] Lorens, C. S., "Theory and Applications of Flow Graphs," Electronics Research Lab., Mass. Inst. Tech., Cambridge, Tech. Rept. No. 317; July, 1957.
[349] Marcus, M. P., "The detection and identification of symmetric switching functions with the use of tables of combinations," IRE Trans. on Electronic Computers, vol. EC-5, pp. 237–239; December, 1956.
[350] McCluskey, E. J., Jr., "Minimization of Boolean functions" *Bell. Sys. Tech. J.*, vol. 35, no. 6, pp. 1417–1444; November, 1956.
[351] McCluskey, E. J., Jr., "Detection of group invariance or total symmetry of a Boolean function," *Bell Sys. Tech. J.*, vol. 35, no. 6, pp. 1445–1466; November, 1956.
[352] Moisil, Gr. C., "Teoria algebrica a mecanismelor automate" (Algebraic theory of automatic devices), 4th Convention of Rumanian Mathematicians, May 27–June 4, 1956. (In Rumanian.)
[353] Moisil, Gr. C., "Teoria algebrica a schemelor de mecanisme automate cu contacte si relee" (Algebraic theory of relay contact networks in automatic devices), *Analele acad. rep. populare Romîne*, Anexa, no. 33; 1956. (In Rumanian.)
[354] Moisil, Gr. C., "Sinteza schemelor cu relee ideale cu ajutorul corpur lor de imaginare ale lui Galois" (Synthesis of circuits with ideal relays with the aid of the Galoisian fields), *Acad. rep.*

populare Romîne, Bul. stiint., Sect. mat., fiz. vol. 8, no. 3, p. 429; 1956. (In Rumanian.)

[355] Moisil, Gr. C., "Intrebuintarea logicilor trivalente in teoria mecanismelor automate. II. Ecuatia caracteristica a unui releu polarizat" (Application of three-symbol logic to the theory of automatic devices. II. Characteristic equation of polarized relay), *Comun. acad. rep. populare Romîne*, vol. 6, no. 3, pp. 388; 1956. (In Rumanian.)

[356] Moisil, Gr. C., "Intrebuintarea logicilor trivalente in teoria mecanismelor automate. III. Scheme cu contacte reale" (Application of three-symbol logic to the theory of automatic devices. III. Circuits with real contacts), *Commun. acad. rep. populare Romîne*, vol. 7, no. 2, pp. 231; 1956. (In Rumanian.)

[357] Moisil, Gr. C., "Intrebuintarea logicilor trivalent in teoria mecanismelor automate. IV. Realizarea functiilor de lucru in functionorea reala" (Application of three-symbol logic to the theory of automatic devices. IV. Realization of operating functions in real action), *Comun. acad. rep. populare Romîne*, vol. 7, no. 8, p. 971; 1956. (In Rumanian.)

[358] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. IV. O teorie trivalenta a releelor polarizate" (Application of Galoisian fields in the theory of automatic devices. IV. Three-symbol theory applied to polarized relays), *Commun. acad. rep. populare Romîne*, vol. 6, no. 4, p. 505; 1956. (In Rumanian.)

[359] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. V. Clasificarea evolutiilor schemelor cu un releu" (Application of Galoisian fields to the theory of automatic devices. V. Classification of the development of single relay circuits), *Comun. acad. rep. populare Romîne*, vol. 6, no. 4, p. 509; 1956. (In Rumanian.)

[360] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. VII. Relee polarizate reale" (Application of Galoisian fields in the theory of automatic devices. VII. Real polarized relays), *Commun. acad. rep. populare Romîne*, vol. 6, no. 5, pp. 618–621; 1956. (In Rumanian.)

[361] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. VIII. Relee basculante reale" (Application of Galoisian fields in the theory of automatic devices. VIII. Real oscillating relays), *Comun. acad. rep. populare Romîne*, vol. 6, no. 5, pp. 625–626; 1956. (In Rumanian.)

[362] Moisil, Gr. C., "Relatiile intre metoda lui Lunt si metoda Tetlin pentru schemele in punte" (Correlation between Lunts and Tsetlins methods for bridge circuits), *Comun. acad. rep. populare Romîne*, vol. 6, no. 6, p. 743; 1956. (In Rumanian.)

[363] Moisil, Gr. C., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automate. IX. Clasificarea schemelor cu un buton si un releu" (Application of Galoisian fields in the theory of automatic devices. IX. Classification of circuits with only one pushbutton and one relay), *Comun. acad. rep. populare Romîne*, vol. 6, no. 9, p. 1055; 1956. (In Rumanian.)

[364] Moisil, Gr. C., and C. Popovici, "Analiza si sinteza schemelor cu comanda directa cu ajutorul imaginarelor lui Galois" (Analysis and synthesis of direct-acting circuits with the aid of Galoisian fields), *Acad. rep. populare Romîne*, Bul. stiint., Sect. mat., fiz., vol. 8, no. 3, p. 455; 1956. (In Rumanian.)

[365] Moore, E. F., and C. E. Shannon, "Reliable circuits using less reliable relays I–II," *J. Franklin Inst.*, vol. 262, no. 3–4, pp. 191–208, 281–298; September–October, 1956. IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-6, p. 137; June, 1957. (Review.) Bell Lab. Monograph no. 2696.

[366] Oberman, R. M. M., "Some analogies between contact, resistor, polar relay and core switching circuits," *Ingenieur*, vol. 68, no. 34, pp. 81–89; August, 1956.

[367] Ostianu, V. M., "Sintez skhem s shagovymi perekljuchateljami" (Synthesis of networks with stepping switches), *Sbornik Rabot Avtomat. i Telemekh.*, AN SSSR, Moscow, pp. 253–267; 1956. (In Russian.)

[368] Petrick, S. R., "A Direct Determination of the Irredundant Forms of a Boolean Function from the Set of Prime Implicants," AF Cambridge Research Center, Bedford, Mass., Tech. Rept. No. 56–110; April, 1956.

[369] Popovici, C., "Asupra teoriei algebrice a functionarii sclipitorilor" (Algebraic theory of mercury relay action), *Comun. acad. rep. populare Romîne*, vol. 6, no. 2, p. 245; 1956. (In Rumanian.)

[370] Povarov, G. N., "Graficheskij sintez simmetricheskikh kontaktnykh skhem" (Graphical synthesis of symmetric contact networks), *Priborostroenie*, no. 12, pp. 7–9; 1956. (In Russian.)

[371] Povarov, G. N., "Sostojanie voprosa o minimal'nom chisle strukturnykh elementov v relejno-kontaktnykh skemakh" (State of the question of the minimum number of structural elements in relay contact networks), in "Telemekh. v narodnom khozjajstve," Moscow, pp. 135–138; 1956. (In Russian.)

[372] Povarov, G. N., "K sintezu simmetricheskikh kontaktnykh skhem" (On the synthesis of symmetric contact networks), *Sbornik Rabot Avtomat. i Telemekh.*, AN SSSR, Moscow, pp. 268–277; 1956. (In Russian.)

[373] Povarov, G. N., "O nekotorykh matrichnykh metodakh analiza relejno-kontaktnykh skhem" (On some matrix methods for analysis of relay contact networks), *Sbornik Rabot Avtomat. i Telemekh.*, AN SSSR, Moscow, pp. 278–285; 1956. (In Russian.)

[374] Povarov, G. N., "O matrichnom analiza svjazej v chastichno orientirovannykh grafakh" (On matrix analysis of the connections in partially oriented graphs), *Uspekhi Mat. Nauk*, vol. 11, no. 5 (71), pp. 195–202; 1956. (In Russian.)

[375] Povarov, G. N., "Do pittanja pro strukturne proektuvannja simmetrichnikh kontaktnykh skhem" (On the investigation of the structural design of symmetric contact networks), *Avtomatika*, (Kiev), no. 4, pp. 48–53; 1956. (In Russian.)

[376] Povarov, G. N., "K matematicheskoj teorij sinteza kontaktnykh (1, k)-poljusnikov" (On the mathematical theory of the synthesis of (1, k)-terminal contact networks), *Dokl. AN SSSR*, vol. 111, no. 1, pp. 102–04; 1956. (In Russian.)

[377] Righi, R., "I metodi matriciali nello studio dei circuiti di commutazione," (Matrix methods in the study of switching circuits), *Ing. ferroviaria*, vol. 11, no. 11, pp. 851–860, November, 1956; no. 12, pp. 963–972; December, 1956. (In Italian.)

[378] Riguet, J., "Algorithmes de Markov et théorie des machines" (Markov algorithms and the theory of machines), *Compt. rend. acad. sci.*, vol. 242, no. 4, pp. 435–437; 1956. (In French.) *J. Symbolic Logic*, vol. 23, p. 62; March, 1958. (Review by G. F. Rose.)

[379] Roginskij, V. N., "Relejnye schetnye skhemy" (Relay computer circuits), in "Telemekh. v narodnom khozjajstve," AN SSSR, Moscow, pp. 139–145; 1956. (In Russian.)

[380] Rohleder, H., "Zur Grundlagen umformung logischer Ausdrücke mit Hilfe programgesteuerter Rechenanlagen" (Transformation of logical equations with the aid of programmed computing machines), *Z. math. Logik Gr. Math.*, vol. 2, pp. 57–58; 1956. (In German.)

[381] Roth, J. P., "A combinatorial topological method for the synthesis of switching systems in n variables," *Bull. Am. Math. Soc.*, vol. 62, p. 183; 1956.

[382] Roth, J. P., "An algorithm for the problem of Quine," *Bull. Am. Math. Soc.*, vol. 62, p. 262; 1956.

[383] Roth, J. P., "A function-space formulation of the switching circuit synthesis problem," *Bull. Am. Math. Soc.*, vol. 62, p. 269; 1956.

[384] Roth, J. P., "Algebraic Topological Methods for the Synthesis of Switching Systems in n Variables," Inst. Advanced Study, Princeton, N. J., no. ECP56-02; April, 1956.

[385] Rouche, N., "Extension aux probabilités du formalisme de l'algèbre logique" (Extension of the logical algebra on formalism probabilities), *Rev. H. F.*, vol. 3, no. 5, pp. 179–182; 1956. (In French.)

[386] Sacerdote, C., "Boolean algebra and electrical circuits," *Illustr. Sci.*, vol. 8, no. 79, pp. 20–23; 1956.

[387] Santesmases, J. G., J. S. Rodriguez, and J. M. Nicolau, "Contribución al estudio de los circuitos de commutación con rectificadores de selenio" (Contribution to the theory of contact networks with selenium rectifiers), *Anales real soc. españ. fiz. y. quim.* vol. 52A, p. 105; 1956. (In Spanish.)

[388] Seshu, S., "On electric circuits and switching circuits," IRE TRANS. ON CIRCUIT THEORY, vol. CT-3, pp. 172–178; September, 1956.

[389] Shestakov, V. I., "Vektorno-algebraicheskij metod analiza i sinteza mnogotaktnykh relejnykh sistem [Rezjume]," (Vector-algebraic analysis and synthesis method for multicycle relay systems [Summary]), 3rd All-Union Conf. Math., AN SSSR, Moscow, pp. 190–191; 1956. (In Russian.)

[390] Sikorski, R., and T. Traczyk, "On some Boolean algebras," *Bull. acad. polon. sci.*, Cl. III, vol. 4, pp. 489–492; 1956.

[391] Sittler, R. W., "Systems analysis of discrete Markov processes," IRE TRANS. ON CIRCUIT THEORY, vol. CT-3, pp. 257–266; December, 1956.

[392] Svoboda, A., "Graphical-mechanical aids for the synthesis of relay circuits," *Nachrtech. Fachber.*, vol. 4, pp. 213–217; 1956. (Original in Czech.) *Ann. télécommun.*, vol. 12, no. 6, p. A471; June, 1957. (Review in French.)

[393] Svoboda, A., "Graficko-mechanické pomucky, uzivané pri analyse kontaktových obvodu" (Graphical-mechanical aids for the synthesis of relay circuits), *Inform. Proc. Mach.*, *Acad. Sci. Prague*, vol. 4, pp. 9–22; 1956. (In Czech.)

[394] Trakhtenbrot, B. A., "Primenenie nekotorykh topologicheskikh invariantov dlja sinteza dvukpoljusnykh kontaktnykh skhem" (Use of certain topological invariants in synthesizing two-terminal contact networks), *3rd All-Union Math. Conf.*, AN SSSR, Moscow, pt. II, p. 136; 1956. (In Russian.)

[395] Urbano, R. H., and R. K. Muller, "A topological method for the determination of minimal forms of Boolean functions,"

IRE Trans. on Electronic Computers, vol. EC-5, pp. 126–132; 1956.
[396] Zemanek, H., "Schaltalgebra" (Switching-circuit algebra), *Nachrtech. Fachber.*, vol. 3, no. 3, pp. 93–113; 1956. (In German.)

Books:

[397] Ashby, W. R., "An Introduction to Cybernetics," John Wiley and Sons, Inc., New York, 295 pp., 26 bibliog.; 1956. IRE Trans. on Electronic Computers, vol. EC-6, p. 307; December, 1957. (Review.)
[398] "Elektronischen Rechenmaschinen und Informationsverarbeitung" (Electronic digital computers and data processing), *Nachrtech. Fachber.*, vol. 4, 237 pp.; 1956. (In German and English.)
[399] Gavrilov, M. A., V. A. Malov, V. A. Zhozhikashvili, *et al.*, "Telemekhanizatsija v narodnom khozjajstve" (Telemechanization in the national economy), Inst. Avtomat. i Telemekh., AN SSSR, 481 pp.; 1956. (In Russian.)
[400] Plechl, O., "Elektromechanische Schaltungen und Schaltgeräte" (Electromechanical contact networks and switching devices), Springer-Verlag, Wien, 224 pp.; 1956. (In German.)
[401] Shannon, C. E., and J. McCarthy, Eds., "Automata Studies," Princeton University Press, Princeton, N. J., 285 pp.; 1956. *J. Assoc. Comput. Mach.*, vol. 3, pp. 376–378, October, 1956. (Review by E. N. Gilbert.)

Dissertations:

[402] Best, D. T., "The Detection of Sneak Paths in Combinational Relay Networks," Moore School of Elec. Engrg., University of Pennsylvania, Philadelphia; 1956.
[403] Cadden, W., "Sequential Circuit Theory," Princeton University Press, Princeton, N. J.; 1956.
[404] Daggett, D. H., "The Analysis of Contact Network Hazards," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 62 pp.; 1956.
[405] Huffman, D. A., "Problems of Sequential Circuit Synthesis," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 33 pp.; 1956.
[406] Jurasov, A. N., "Nekotorie analiticheskie metody sostarlenija i preobrazovanija relejno-kontaktnykh skhem" (Some analytical methods for designing and transforming of relay contact networks), V.Z.P.I.; 1956. (In Russian.)
[407] McCluskey, E. J., Jr., "Algebraic Minimization and the Design of Two-Terminal Contact Networks," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 135 pp.; 1956.
[408] O'Loughlin, J. B., "Multiple-Ground Coding in Iterative Switching Circuits." Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 62 pp.; 1956.
[409] Tsukanov, T. T., "Matrichnij analisator relejno-kontaktnykh skhem i voprosy egoprimenenija" (Matrix analysator for relay contact networks and the problems in using it), L.I.I.Zh.D.T.; 1956. (In Russian.) (See also [320] and [339].)

1957

[410] Ackerman, S. S., "Symbolic logic: summary of subject and its application to industrial engineering," *J. Ind. Engrg.*, vol. 8, no. 5, pp. 293–299; September–October, 1957.
[411] Acred, N. B., "Control circuit design," *Electronic Engrg.*, vol. 29, pp. 586–590; December, 1957.
[412] Akers, S. B., "On a theory of Boolean functions," *Bull. Am. Math. Soc.*, vol. 63, p. 350; November, 1957.
[413] Arkhangel'skaia, A. A., V. F. D'jachenko, and V. G. Lazarev, "Primenenie teorii relejno-kontaktnykh skhem pri analize i sinteze telefonykh skhem" (Application of the relay contact network theory to analysis and synthesis of telephone circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 64; 1957. (In Russian.)
[414] Arkhangel'skaia, A. A., V. G. Lazarev, and V. N. Roginskij, "Mashinizatsija protsessa sinteza relejnykh skhem" (Mechanization of the synthesis process of relay circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 73; 1957. (In Russian.)
[415] Ashenhurst, R. L., "The decomposition of switching functions," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 74–116; April, 1957.
[416] Aufenkamp, D. D., and F. E. Hohn, "Analysis of sequential machines," IRE Trans. on Electronic Computers, vol. EC-6, pp. 276–285; December, 1957.
[417] Badillo-Barallat, M. C., "Logica trivalente en la automatizacon de los circuitos" (Three-valued logic in the circuit automatisation), *Rev. calc. autom. cibern.*, no. 16, pp. 1–7; 1957. (In Spanish.)
[418] Belevitch, V., "Some relations between the theory of contact networks and conventional network theory," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 3–12; April, 1957.
[419] Bernard, E., "Les dispositifs statiques de commutation" (Static considerations of electric switching networks), *Automatisme*, vol. 2, no. 8, pp. 299–304; 1957. (In French.)
[420] Blokh, A.Sh., "Sintez relejno-kontaktnykh skhem" (Synthesis of relay contact networks), *Dokl. AN SSSR*, vol. 117, no. 4, pp. 609–612; 1957. (In Russian.)
[421] Blokh, A. Sh., "Spetsial'nye slutsai sinteza kontaktnykh (p,q)-poljusnikov" (Special cases of the synthesis of (p,q)-terminal contact networks), *Trans. Minsk Higher Radiotech. Engrg. School*, vol. 7, pp. 57–63; 1957. (In Russian.)
[422] Blokh, A.Sh., "O Bulevykh funktsijakh" (On Boolean functions), *Trans. Minsk Higher Radiotech. Engrg. School*, vol. 6, pp. 66–71; 1957. (In Russian.)
[423] Brooks, R. W., "Logical design with symbolic logic," *Instr. and Automation*, vol. 30, no. 3, pp. 457–463; 1957.
[424] Burks, A. W., "The logic of fixed and growing automata," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 147–188; April, 1957.
[425] Calingaert, F., "Multiple-output relay switching circuits," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. II, pp. 59–73; April, 1957. Dissertation, Harvard University; May, 1955.
[426] Campeau, J. O., "The synthesis and analysis of digital systems by Boolean matrices," IRE Trans. on Electronic Computers, vol. EC-6, pp. 230–241; December, 1957.
[427] Cardot, C., "Application de la théorie des treillis distributifs à l'étude de la distribution des programmes par télécommandes" (Application of distribution lattice theory to the study of the distribution of remote-control programming), *Rev. gén. élec.* vol. 66, no. 1, pp. 27–39; January, 1957. (In French.)
[428] Constantinescu, P., "O sinteze kontaktnykh mnogopoljusnikov s ventil'nymi elementami" (Synthesis of contact multiterminal networks with rectifying elements), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 23; 1957. (In Russian.)
[429] Davies, D. W., "Switching functions of three variables," IRE Trans. on Electronic Computers, vol. EC-4, pp. 265–275; December, 1957.
[430] Diamand, M., "Circuitos logicos" (Logical circuits), *Rev. telegr. electrón.*, vol. 46, no. 539, pp. 482–486; 1957. (In Spanish.)
[431] Eguchi, S., (Transformation of relay networks), *J. Inst. Elec. Commun. Engrs. Japan*, vol. 40, no. 3, pp. 229–235; March, 1957. (Original in Japanese.)
[432] Erbacher, W., (Analytical treatment of electric networks with special reference to matrices), *Österr. Z. Elekt. wirtsch.*, vol. 10, no. 12, pp. 435–441; December, 1957. (Original in German.)
[433] Finikov, B. I., "Ob odnom semejstve klassov funktsii algebry logiki i ikh realizatsii v klasse P-skhem" (On one particular family of classes of functions in the algebra of logic and its realization in a class of P networks [Series-parallel]), *Dokl. AN SSSR*, vol. 115, no. 2, pp. 247–248; July, 1957. (In Russian.) *Automation Express*, vol. 1, no. 1, pp. 15–16; May, 1958. (Review.)
[434] Gádor, L., "Some relations in the field of relay algebra," *Elektrotechnica*, vol. 50, no. 4, pp. 138–144; April, 1957.
[435] Gavrilov, M. A., "Postroenie relejnykh skhem s mostikovymi soedinenijami" (Setting up relay circuits with bridge connections), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 24; 1957. (In Russian.)
[436] Ghazala, M. J., "Irredundant disjunctions and conjunction forms of a Boolean function," IBM *J. Res. and Dev.*, vol. 1, no. 2, pp. 171–176; April, 1957.
[437] Gould, R., "The Application of Graph Theory to the Synthesis of Contact Networks," Bell Labs. Rept. No. 18; 1957. Dissertation, Harvard University, Cambridge, Mass.; 1957.
[438] Gould, R., "The application of graph theory to the synthesis of contact networks," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 244–292; April, 1957.
[439] Grebenshchikov, V. N., "Metod sinteza mnogopoljusnykh relejno-kontaktnykh skhem" (Method of synthesis of multiterminal relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 49; 1957. (In Russian.)
[440] Harris, B., "An algorithm for determining minimal representation of a logic function," IRE Trans. on Electronic Computers, vol. EC-6, p. 103; June, 1957.
[441] Higonnet, R. A., and R. A. Gréa, "Some aspects of switching algebra," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 281–284; April, 1957.
[442] Hohn, F. E., "2*N*-terminal contact networks," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 51–58; April, 1957.
[443] Hohn, F. E., S. Seshu, and D. D. Aufenkamp, "The theory of

nets," IRE Trans. on Electronic Computers, vol. EC-6, pp. 154–161; September, 1957.
[444] Holbrook, B. D., "Some logical requirements for the control of switching networks," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 235–240; April, 1957. Bell Lab. Monograph no. 3520.
[445] Huffman, D. A., "The design and use of hazard-free switching networks," *J. Assoc. Comput. Mach.*, vol. 4, no. 1, pp. 47–62; January, 1957.
[446] Huffman, D. A., "An algebra for periodically time-varying linear binary sequence transducers," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 189–203; April, 1957.
[447] Ioanin, G., "O skhemakh, rabotajushchikh na rele s real'nymi kontaktami" (Circuits operating with actual relay contacts), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, pp. 56–58; 1957. (In Russian.)
[448] Jablonskij, S. V., "Ob odnom semejstve klassov funktsij algebry logiki, doruskajushchikh prostuju skhemnuju realizatsiju" (A special family of classes of mathematical logic functions which is realizable by simple circuits), *Uspekhi Mat. Nauk*, vol. 6, pp. 186–199; 1957. (In Russian.)
[449] Jurasov, A. N., "Sintez mnogotaktnykh skhem, iskhodjashchij iz periodov vkljuchenija i otkljuchenija" (Synthesis of multicycle circuits based on the connecting and disconnecting periods), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 65; 1957. (In Russian.)
[450] Kantorovich, L. V., "Ob odnom matematicheskoj simbolike, udobnoj dlja provedenija vychislenij na mashinakh" (Mathematical notation convenient for applications in computer calculations), *Dokl. AN SSSR*, vol. 113, no. 4, pp. 738–741; 1957. (In Russian.)
[451] Kharkevich, A. D., "Kommutatsionnye skhemy" (Commutation circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 36; 1957. (In Russian.)
[452] Kharkevich, A. D., "Primenenie metoda verojatnostnykh grafov dlja analiza kommutatsionnykh skhem" (Application of probability-graph method in the analysis of commutation circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 44; 1957. (In Russian.)
[453] Kjellberg, G., "Logical and other kinds of independence," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 117–124; April, 1957.
[454] Klein, M., F. Williams, and H. Morgan, "Two-terminal relay circuits," *Instr. and Automation*, vol. 20, no. 1, pp. 71–73; January, 1957.
[455] Kurepa, G., "Sets—logics—machines," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass. pt. 1, pp. 137–146; April, 1957.
[456] Kuznetsov, A. V., "Nekotorye voprosy matematicheskoj teorii kontaktnykh skhem" (Some problems of the mathematical theory of contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 33; 1957. (In Russian.)
[457] Lazarev, V. G., "Metodika opredelenija minimal'nogo chisla rele, neobkhodimykh dlja postroenija relejnoj skhemy po zadannym uslovijam" (Method for determining the minimum number of relays required for setting up relay circuits according to given premises), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 38; 1957. (In Russian.)
[458] Lipp, J. P., "Topology of switching elements versus reliability," IRE Trans. on Reliability and Quality Control, vol. RQC-10, pp. 21–33; June, 1957.
[459] Lunts, A. G., "Metod sinteza (1,k)-poljusnika" (Method of synthesis of (1,k)-terminal networks), *Dokl. AN SSSR*, vol. 112, no. 1, pp. 35–37; 1957. (In Russian.) *J. Symbolic Logic* vol. 22, no. 4, p. 378; December, 1957. (Review.) *Automation Express*, vol. 1, no. 1, pp. 9–10; May, 1958. (Review.)
[460] Lunts, A. G., "Matrichnyj metod i metod kharakteristicheskikh funktsij v teorii kontaktnykh skhem" (Matrix method and the method of characteristic functions in the theory of contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR; 1957. (In Russian.)
[461] Maistrova, T. L., "Primenenie implikatsii v teorii relejnykh skhem" (Application of implicates in the theory of relay contact networks), in collection of articles V.Z.P.I., no. 16, pp. 47–50; 1957. (In Russian.)
[462] Maistrova, T. L., "Primenenie mnogoznachnoj logiki v teorii relejno-kontaktnykh skhem" (Application of multivalued logic to the theory of relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 52; 1957. (In Russian.)
[463] Marcus, M. P., "Minimization of the partially-developed transfer tree," IRE Trans. on Electronic Computers, vol. EC-6, pp. 92–95; June, 1957.
[464] Marcus, M. P., "Minimizing partially developed relay tree circuits," *Elec. Mf.*, vol. 60, no. 2, pp. 102–105; August, 1957.
[465] Markov, A. A., "Matematicheskaja logika i vychislitel'naja matematika" (Mathematical logic and numerical calculations), *Vestnik AN SSSR*, no. 8, pp. 21–23; 1957. (In Russian.)
[466] Markov, A. A., "Ob inversionnoj slozhnosti sistemy funktsij" (On an inversion complex of a system of functions), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 13; 1957. (In Russian.)
[467] Markov, A. A., "Ob inversionnoj slozhnosti sistemy funktsij" (On an inversion complex of a system of functions), *Dokl. AN SSSR*, vol. 116, no. 6, pp. 917–919; 1957. (In Russian.)
[468] Mekler, Ja. I., "Konstituenty, dostatochnye dlja obespechenija rabot relejno-kontaktnykh skhem" (Components sufficient to guarantee operation of a relay contact network), *Dokl. AN SSSR*, vol. 117, no. 4, pp. 613–615; December, 1957. (In Russian.) *Automation Express*, vol. 1, no. 1, p. 15; May, 1958. (Review.)
[469] Mekler, Ja. I., "Perekhodnye rezhimy v relejno-kontaktnykh skhem" (Transient processes in relay contact networks), *Avtomat. i Telemekh.*, vol. 18, no. 1, pp. 59–70; 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 1, pp. 63–74; 1957. (Transl.)
[470] Mekler, Ja. I., "Graficheskij metod postroenija relejno-kontaktnykh skhem" (Graphical method of setting up relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, pp. 53–55; 1957. (In Russian.)
[471] Miehle, W., and A. Burroughs, "Truth-function evaluation," *J. Assoc. Comput. Mach.*, vol. 4, no. 2, pp. 189–92; 1957.
[472] Moisil, Gr. C., "Algebraicheskaja teorija raboty real'nykh relejno-kontaktnykh skhem" (Algebraic theory of the operations of actual relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 12; 1957. (In Russian.)
[473] Moisil, Gr. C., "Razvitie teorii relejno-kontaktnykh skhem v Rum'nskoj Narodnoj Respublike" (Development of the theory of relay contact networks in the Rumanian People's Republic), *All-Union Moscow Conf. Theory Relay Oper. Devices*, AN SSSR, p. 8; 1957. (In Russian.)
[474] Moisil, Gr. C., "Sur la synthèse des schemas à relais polarisés" (On the synthesis of circuits with polarized relays), *Bull. Math. Inst. Bulgar. Acad. Sci.*, vol. 2, no. 2, p. 121; 1957. (In French.)
[475] Muller, D. E., "A theory of asynchronous circuits," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 204–243; April, 1957.
[476] Nolin, L., "Algèbre de Boole et calcul des propositions" (Boolean algebra and the calculus of propositions), *Compt. rend. acad. sci. France*, vol. 244, no. 15, pp. 1999–2002; April, 1957. (In French.)
[477] Novais, J. C. V., "Introducas ces algebras de Boole" (Introduction to Boolean algebra), *Gaz. matem. (Lisbon)*, no. 66–67, pp. 1–8; March–June, 1957. (In Portuguese)
[478] Ostianu, V. M., "Sintez skhem s mnogopozitsionnymi elementami" (Synthesis of circuits with multipositional elements), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 48; 1957. (In Russian.)
[479] Ostianu, V. M., and Ju. L. Tomfel'd, "Ob odnom primenenii matematicheskoj logiki" (On a particular application of mathematical logic), *Sci. Notes Kishinev State Univ., Phys.-mat.*, vol. 29, pp. 227–233; 1957. (In Russian.)
[480] Paleček, J., "Application of relay algebra to the solution of circuits in communication systems," *Slaboproudý obzor.*, vol. 18, no. 7, pp. 490–495; 1957. (Original in Czech.)
[481] Parkhomenko, P. P., "Avtomatizatsija protsessov analiza relejnykh skhem" (Automation of the process of analyzing relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 70; 1957. (In Russian.)
[482] Piesch, J., "Beitrage zur Modernen Schaltalgebra" (Contributions to the modern algebra of switching circuits), *Sci. Elec.*, vol. 3, no. 1, pp. 16–25; 1957. (In German.)
[483] Poljak, A., "Releovy Retezy," (Relay chains), *Slaboproudý obzor.*, vol. 8, no. 8, pp. 542–545; 1957. (In Czech.)
[484] Popovich, K., "Minimal'naja diz'junktivnaja forma Bulevykh funktsii" (Minimal disjunctive form of a Boolean function), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 21; 1957. (In Russian.)
[485] Povarov, G. N., "Absoljutno zhestkie Markovskie tsepi v avtonomnykh perekljuchatel'nykh ustrojstvakh" (Absolutely rigid Markov circuits in independent switching devices), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 68; 1957. (In Russian.)
[486] Povarov, G. N., "Nekotorye obshchie voprosy teorii perekljuchenija" (Some general questions of the theory of switching), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 29; 1957. (In Russian.)
[487] Povarov, G. N., "Metod sinteza vychislitel'nykh i upravljajushchikh kontaktnykh skhem" (Method of synthesis of com-

puting and controlling circuits), *Avtomat. i Telemekh.*, vol. 18, no. 2, pp. 145–162; 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 2, pp. 159–178; 1957. (Transl.) *J. Symbolic Logic*, vol. 23, pp. 61–62; March, 1958. (Review.)

[488] Povarov, G. N., "A mathematical theory for the synthesis of contact networks with one input and 'k' outputs," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 74–94; April, 1957.

[489] Prim, R. C., "Shortest connection networks and some generalizations," *Bell Sys. Tech. J.*, vol. 36, no. 6, pp. 1389–1401; November, 1957. Bell Lab. Monograph no. 2908.

[490] Rodin, V. M., "Elektronnij analizator kontaktnykh skhem" (Electronic analyzer of contact networks), *Avtomat. i Telemekh.*, vol. 18, no. 5, pp. 437–443; 1957. (In Russian.)

[491] Roginskij, V. N., "Ravnosil'nye preobrazovanija relejnykh skhem" (Equivalent transformation of relay circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 27; 1957. (In Russian.)

[492] Roginskij, V. N., "Graficheskij metod postroenija kontaktnykh (1,k)-poljusnikov" (Graphical method of setting up (1,k)-terminal contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 27; 1957. (In Russian.)

[493] Roginskij, V. N., "Graficheskij metod sinteza kontaktnykh skhem" (Graphical method of synthesizing contact networks), *Elektrosviaz'*, no. 11; 1957. (In Russian.)

[494] Roginskij, V. N., "Ravnosil'nye preobrazovanija relejnykh skhem klassa P" (Equivalent transformations of relay networks of class P [series-parallel]), *Dokl. AN SSSR*, vol. 113, no. 2; 1957. (In Russian.)

[495] Roginskij, V. N., "A graphical method for the synthesis of multiterminal contact networks," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 302–315; April, 1957.

[496] Roginskij, V. N., "Sintez smeshannikh relejnykh skhem klassa P" (Synthesis of mixed relay circuits of class P [series-parallel]), *Avtomat. i Telemekh.*, vol. 18, no. 12, pp. 1120–1131; 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 12, pp. 1165–1174; 1957. (Transl.) *Automation Express*, vol. 1, no. 1, p. 16; May, 1958. (Excerpt.)

[497] Rohleder, H., "Über eine Theorie einiger Klassen von elektrischen Schaltungen" (On a theory of some of the classes of electrical switching circuits), *Z. math. Logik Gr. Math.*, vol. 3, pp. 225–291; 1957. (In German.)

[498] Roth, J. P., "Algebraic topological methods in synthesis," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 57–73; April, 1957.

[499] Rubinoff, M., "Remarks on the design of sequential circuits," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 241–280; April, 1957.

[500] Ryser, H. J., "Combinational properties of matrices of zeros and ones," *Can. J. Math.*, vol. 9, no. 3; 1957.

[501] Scheinman, A. H., "A numerical-graphical method for synthesizing switching circuits," *Trans. AIEE, (Commun. and Electronics)*, vol. 76, no. 34, pp. 687–689; January, 1958. Bell Lab. Monograph no. 2979.

[502] Sebestyen, G., "A design technique for pedestal-free switching circuits," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-6, pp. 162–166; September, 1957.

[503] Semon, W., "Matrix methods in the theory of switching," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 13–50; April, 1957.

[504] Seshu, S., and F. E. Hohn, "Symmetric polynomials in Boolean algebras," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 225–234; April, 1957.

[505] Shestakov, V. I., "Algebraicheskij metod sinteza mnogotaktnykh sistem *R*-pozitsionnykh rele," (Algebraic method of synthesis of multicycle system with *R*-positional switches), *Dokl. AN SSSR*, vol. 112, no. 1, pp. 62–65; 1957. (In Russian.)

[506] Shestakov, V. I., "Ob algebraicheskom metode analiza i sinteza mnogotaktnykh relejnykh skhem" (Algebraic method of analyzing and synthesizing multicycle relay circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 16; 1957. (In Russian.)

[507] Shnarevich, D. I., "Preobrazovanie relejnykh skhem s dopolnitel'nymi obmotkami" (Transformation of relay circuits with additional windings), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, pp. 62–63; 1957. (In Russian.)

[508] Shnarevich, D. I., "Analiticheskie metody preobrazovanija relejnykh skhem" (Analytical method of transforming relay circuits), *19th Sci. Tech. Conf. Leningrad Inst. Railway Transport Engrs.*, pp. 7–10; 1957. (In Russian.)

[509] Singer, T., "Some uses of truth tables," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 125–133; April, 1957.

[510] Svoboda, A., "Nekotorye primenenija kontaktnykh setok," (Some applications of contact circuits), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 10; 1957. (In Russian.)

[511] Svoboda, A., "Some applications of contact grids," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 293–305; April, 1957.

[512] Svoboda, F., "Užití neurčité dvouhodnotové Booleovy funkce na synthesu jednotakních hradlových schemat" (Use of indeterminate two-valued Boolean functions in synthesizing switching circuits), *Stroje na Zpracování informací*, vol. 2, pp. 209–224; 1954. (In Czech.) *J. Symbolic Logic*, vol. 22, pp. 99–100; March, 1957. (Review.)

[513] Svoboda, F., "K teorii sinteza kontaktnykh skhem" (Theory for the synthesis of contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 20; 1957. (In Russian.)

[514] Svoboda, F., "Sintez relejnykh skhem pri pomoschi mashin" (Synthesis of relay networks by means of machines), *Avtomat. i Telemekh.*, vol. 18, no. 3, pp. 240–255; 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 3, pp. 265–279; 1957. (Transl.) *J. Symbolic Logic*, vol. 23, p. 61; March, 1958. (Review.)

[515] Tezuka, Y., (On sequential switching circuits), *J. Inst. Elec. Commun. Engrs. Japan*, vol. 40, no. 1, pp. 23–29; January, 1957. (Original in Japanese.)

[516] Trakhtenbrot, B. A., "Ob operatorakh realizuemykh v logicheskikh setjakh" (On operators realizable in logical nets), *Dokl. AN SSSR*, vol. 112, no. 2, pp. 1005–1007; 1957. (In (Russian.)

[517] Tsetlin, M. L., "Matrichnyj metod analiza i sinteza elektronno-impul'snykh i relejno-kontaktnykh [neprimitivnykh] skhem" (Matrix method for analysis and synthesis of electronic-pulse and relay contact [nonprimitive] networks), *Dokl. AN SSSR*, vol. 117, no. 6; 1957. (In Russian.)

[518] Tsukanov, T. T., "Matrichnyj analizator relejno-kontaktnykh skhem" (Matrix analyzer of relay contact networks), *All-Union Moscow Conf. Theory Relay Operating Devices*, AN SSSR, p. 72; 1957 (In Russian.)

[519] Unger, S. H., "A Study of Asynchronous Logical Feedback Networks," Electronics Research Lab., Mass. Inst. Tech., Cambridge, Tech. Rept. No. 320, 45 pp.; 1957.

[520] Weinberg, L., and P. Slepian, "Series-parallel networks," IRE TRANS. ON CIRCUIT THEORY, vol. CT-4, pp. 226–227; September, 1957.

[521] Weinberger, A., and H. Loberman, "Symbolic designations for electrical connections," *J. Assoc. Comput. Mach.*, vol. 4, no. 4, pp. 420–427; 1957.

[522] Zapletin, N. V., "Matrichnyj metod analiza i sinteza relejno-kontaktnykh skhem" (Matrix method for the analysis and synthesis of relay contact networks), *Collection Leningrad Electrotech. Inst.*, vol. 3, no. 33, pp. 143–150; 1957. (In Russian.)

[523] Zemanek, H., "Einführung in die Schaltalgebra" (Introduction to the algebra of switching circuits), *Tech. Schriftenreihe, Inform. Dienst*, Siemens-Halske Gesellsch., Wiener Schwachstrome Werke, Vienna; July, 1957.

[524] Zemanek, H., "Die Lösungen von Gleichungen in der Schaltalgebra" (Solving the equations in the algebra of switching circuits), *Arch. Elektrischen Übertr.*, vol. 11, no. 11; 1957. (In (German.)

Books:

[525] "Tezisy dokladov na vsesojuznom soveshchanii po teorii relejnykh skhem" (Thesis of the Papers read at the All-Union Moscow Conference on the Theory of Relay Operating Devices), AN SSSR, Moscow; 1957. (In Russian.)

Dissertations:

[526] Jenney, R. F., "A Topological Study of Switching Circuit Memory Requirements," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 71 pp.; 1957.

[527] Kellner, W. G., "Linear Planar Iterative Switching Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 66 pp.; 1957.

[528] Mullin, A. A., "Probabilistic Logics in the Synthesis of Reliable Sequential Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 104 pp.; 1957.

[529] Roth, C. H., "Digital Computer Solution of Switching Circuit Problems," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 165 pp.; 1957.

[530] Unger, S. H., "A Study of Asynchronous Logical Feedback Networks," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 79 pp.; 1957. (See also [425] and [427].)

1958

[531] Abhyankar, S., "Minimal sum of products of sum expression of Boolean functions," IRE Trans. on Electronic Computers, vol. EC-7, pp. 268–276; December, 1958.

[532] Arango, H., and I. Santos, "The operation 'bridge' in Boolean algebra," *Cienc. y Tecnol.*, no. 125, pp. 168–175; 1958.

[533] Aufenkamp, D. D., "Analysis of sequential machines II," IRE Trans. on Electronic Computers, vol. EC-7, pp. 299–306; December, 1958. (See also [558].)

[534] Badillo-Barallat, M. C., "Nuevas posibilidades dentro de la logica trivalente y problemas resueltos con reles y redes" (New possibilities in the three-valued logic and the problems of relay algebra), *Calc. aut. y cib.*, no. 19, pp. 28–39; 1958. (In Spanish.)

[535] Beatson, T. J., "Component minimization in electronic switching," *Electronic Engrg.*, vol. 77, no. 3, p. 213; 1958.

[536] Beatson, T. J., "Minimization of components in electronic switching circuits," *Trans. AIEE, (Commun. and Electronics)*, vol. 77, no. 37, pp. 283–291; July, 1958.

[537] Beizer, B., "Extension of Boolean algebra for analysis of mixed switch diode circuits," Proc. IRE, vol. 46, pp. 779–780; April, 1958.

[538] Beizer, B., and S. M. Leibholz, "Designing sequential circuits," *Elec. Mf.*, vol. 62, no. 3, pp. 67–69, 298, 300; 1958

[539] Beizer, B., and S. M. Leibholz, "Engineering applications of Boolean algebra," *Elec. Mf.*, vol. 61, no. 5, pp. 129–138, May, 1958; no. 6, pp. 98–108, 326, 328, June, 1958; vol. 62, no. 1, pp. 100–109, July, 1958; no. 2, pp. 108–117, 272, August, 1958; no. 3, pp. 67–79, 298, 300, September, 1958.

[540] Bellomi, C., "Le dirette unioni orizzontali nell algebra dei circuiti variabili" (Direct horizontal combinations in the algebra of switching circuits), *Ing. ferroviaria*, vol. 13, no. 2, pp. 115–130; February, 1958. (In Italian.)

[541] Berlin, R. D., "Synthesis of N-valued switching circuits," IRE Trans. on Electronic Computers, vol. EC-7, pp. 52–56; March, 1958.

[542] Blokh, A. Sh., "Sintez skhem s shagovymi perekljuchateljami" (Synthesis of circuits with stepping switches), *Trudy Minsk. Vyssh.*, 12, pp. 75–79; 1958. (In Russian.)

[543] Boll, M., "La logique dite propositionelle et la commutation," (Logic of propositions and switching circuits), *Automatisme*, vol. 3, no. 3, pp. 95–101; 1958. (In French.)

[544] Brown, D. M., "Boolean algebra, ands, ors and nots, among other things," *Am. Math. Monthly*, vol. 65, no. 8, p. 659; 1958.

[545] Byerly, R. T., R. W. Long, and C. W. King, "Logic for applying topological methods to electric networks," *Trans. AIEE, (Commun. and Electronics)*, vol. 77, no. 39, pp. 657–667); November, 1958.

[546] Calderwood, J. H., and A. Porter, "Pattern recognition in the synthesis of complex switching systems," *J. Electronic Contr.*, vol. 4, no. 5, pp. 466–480; May, 1958.

[547] Copi, I. M., C. C. Elgot, and J. B. Wright, "Realization of events by logical nets," *J. Assoc. Comput. Mach.*, vol. 5, no. 2, pp. 181–196; April, 1958.

[548] Corio, L. J., "Boolean algebra," *Western Elec. Engr.*, vol. 2, no. 3, pp. 46–48; July, 1958.

[549] Diamand, M., "Circuitos logicos. IV. Descripcion del funcionamiento del dispositivo electronico" (Logical circuits. IV. Description of the operation of electronic devices), *Rev. telegr. electrón.*, vol. 46, no. 545, pp. 130–133 and 149; March, 1958. (In Spanish.)

[550] Dickinson, W. E., and R. M. Walker, "Reliability improvement by the use of multiple element switching circuits," *IBM J. Res. and Dev.*, vol. 2, pp. 142–147; April, 1958.

[551] Eguchi, S., (A new method of assigning secondary relays in sequential switching circuits), *J. Inst. Elec. Commun. Engrs. Japan*, vol. 41, no. 4, pp. 475–481; 1958. (Original in Japanese.)

[552] Epstein, G., "Synthesis of electronic circuits for symmetric functions," IRE Trans. on Electronic Computers, vol. EC-7, pp. 57–60; March, 1958.

[553] Fitch, F. B., "Representation of sequential circuits in combinatory logic," *Phil. Sci.*, vol. 25, pp. 263–279; October, 1958.

[554] Flehinger, B. J., "Reliability improvement through redundancy at various system levels," *IBM J. Res. and Dev.*, vol. 2, no. 2, pp. 148–158; IRE National Convention Record, vol. 6, pt. 6, pp. 137–151; 1958.

[555] García, L. C., (The method of matrices or grids in the design of circuits with bi-valent elements), *Rev. Telecomun.*, vol. 13, pp. 15–26; December, 1958. (Original in Spanish.)

[556] Gavrilov, M. A., V. M. Ostianu, V. N. Rodin, and B. L. Timoteev, "Realizatsija skhem diskretnikh korrektorov" (Realization of discrete correcting schemes), *Dokl. AN SSSR*, vol. 123, no. 6, pp. 1025–1028; 1958. (In Russian.)

[557] Gilbert, E. N., "Gray codes and paths on the n-cube," *Bell Sys. Tech. J.*, vol. 37, pp. 815–826; May, 1958.

[558] Gillespie, R. G., and D. D. Aufenkamp, "On the analysis of sequential machines," IRE Trans. on Electronic Computers vol. EC-7, pp. 119–122; June. 1958. (See also [533].)

[559] Glattli, H. Z., "Systematische Herleitung logischer Schaltungen" (Systematic development of logical switching circuits), *Z. angew. Math. u. Phys.*, vol. 9a, no. 3, pp. 288–289; 1958. (In German.)

[560] Gould, R., "A note on contact networks for switching functions of four variables," IRE Trans. on Electronic Computers, vol. EC-7, pp. 195–198; September, 1958.

[561] Grebenshchikov, V. N., "Metod sinteza mnogopoljasnykh kontaktnykh skhem" (Method of synthesizing multiterminal contact networks), *Dokl. AN SSSR*, vol. 119, no. 2, pp. 278–281; March, 1958. (In Russian.) *Soviet Physic Dokl.*, vol. 3, pp. 229–233; March–April, 1958. (Transl.)

[562] Grebenshchikov, V. N., "Metod sinteza mnogopoljasnykh kontaktnykh skhem" (Method of synthesizing multiterminal contact networks), *Vestnik Moscow State Univ.*, no. 3, pp. 117–127; November, 1958. (In Russian.) *Automation Express*, vol. 1, no. 9, pp. 1–5; May, 1959. (Transl.)

[563] Greniewski, M., [(m+n)-element algebras and their applications to relay contact networks], *Zastow. Mat.*, vol. IV, pp. 142–168; 1958. (Original in Rumanian.)

[564] Grisamore, N. T., L. S. Rotolo, and G. U. Uyehara, "Logical design using the Stroke function," IRE Trans. on Electronic Computers, vol. EC-7, pp. 181–183; June, 1958.

[565] Hirschhorn, E., "Simplification of a class of Boolean functions," *J. Assoc. Comput. Mach.*, vol. 5, pp. 67–75; January, 1958.

[566] Huffman, D. A., "Solvability Criterion for Simultaneous Logical Equations," Electronics Research Lab., Mass. Inst. Tech., Cambridge, Quart. Progress Rept., pp. 87–88; January, 1958.

[567] Huffman, D. A., "Problems of Sequential Circuit Synthesis," Session Advanced Theory Logical Design Digital Computers, University of Michigan, Ann Arbor; June, 1958.

[568] Huzino, S., "On some sequential machines and experiments," *Mem. Fac. Sci. Kyushu Univ.*, Fukuoka, Ser. A, vol. 12, no. 2, pp. 136–158; 1958.

[569] Huzino, S., "Reduction theorems on sequential machines," *Mem. Fac. Sci. Kyushu Univ.*, Fukuoka, Ser. A, vol. 12, no. 2, pp. 159–179; 1958.

[570] Ioanin, G., "Sur un type de problemes concernant les schèmas à selecteurs" (On one type of problems concerning selector switching circuits), *Acta Logica*, no. 1, p. 187; 1958. (In French.)

[571] Ioanin, G., "Sintez skhem s shagovymi perekljuchateljami" (Synthesis of systems with stepping selector switches), *Avtomat. i Telemekh.*, vol. 19, no. 9, pp. 855–863, September, 1958. (In Russian.) *Automation and Remote Control*, vol. 19, no. 9, pp. 835–843; September, 1958. (Transl.)

[572] Jablonskij, S. V., "Funktsionalnie postroenija v k-znachoj logike" (Functional construction in k-valued logics), *Trudy Inst. Mat.*, vol. 51, 1958. (In Russian.) Izd. AN SSSR, Moscow, pp. 5–142; 1958. (Book.)

[573] Jablonskij, S. V., "O predel'nykh logikakh" (On threshold logic), *Dokl. AN SSSR*, vol. 118, no. 4, pp. 657–660; 1958. (In Russian.)

[574] Jakovlev, S. M., "Teoriia struktur kombinatornykh mekhanizmov" (Structural theory of combinational mechanisms), *Avtomat i Telemekh.*, vol. 19, no. 3, pp. 221–227; 1958. (In Russian.) *Automation and Remote Control*, vol. 19, no. 3, pp. 221–227; 1958. (Transl.)

[575] Kaltenecker, H., "Einiges über Schaltglieder, Logikelemente und Kontaktalgebra" (On switching devices, logical elements and contact algebra), *Arch. Tech. Messen.*, no. 267, pp. R49–R54; April, 1958. (In German.)

[576] Karpova, N. A., "O kontaktnykh skhemakh dlja monotonnykh funktsij" (Contact networks for monotone functions), *Dokl. AN SSSR*, vol. 123, no. 1, pp. 25–27; 1958. (In Russian.)

[577] Klir, J., and L. Seidl, "Metody analisy a syntesy reléových obvodu. 2. Čast: Metody vyvimete v ČSR" (Methods of analysis and synthesis of relay contact networks. 2. part: Methods applied to the ČSR), *Slaboproudý obzor*, vol. 19, no. 9, pp. 596–605; 1958. (In Czech.)

[578] Kreysa, K., "Matematicka metoda řešeni reléových shémat pohona" (Mathematical methods for relay contact-network chains), *Automatisace*, no. 12, pp. 398–404; 1958. (In Czech.)

[579] Kuznetsov, A. V., "Ob odnom svojstve funktsij, realizuemykh neploskimi bespovtornymi skhemani" (On a particular property of functions realized by nonplanar, nonrepetitive circuits), in (Collected papers on mathematical logic and its applications to a certain question of cybernetics,) Izd. AN SSSR, Moscow, pp. 174–185; 1958. (In Russian.) *Trudy V. A. Steklov Mat. Inst.*, (51).

[580] Kuznetsov, A. V., "O bespovtornykh kontaktnykh skhemakh i bespovtornykh superpozitsijakh funktsij algebry logiki" (On

nonrepetitive contact networks and nonrepetitive superposition of the functions in the algebras of logic), in (Collected papers on mathematical logic and its applications to a certain question of cybernetics), Izd. AN SSSR, Moscow, pp. 186–225; 1958. (In Russian.) *Trudy V. A. Steklov Mat. Inst.*, (51).

[581] Lazarev, V. G., and I. L. Sagalovich, "Ob odnom tipe kommutatsionnykh skhem" (On a special type of switching circuits), *Avtomat. i Telemekh.*, vol. 19, no. 5, pp. 464–467; May, 1958. (In Russian.) *Automation and Remote Control*, vol. 19, no. 5, pp. 457–460; May, 1958. (Transl.)

[582] Löfgren, L., "Automata of high complexity and methods of increasing their reliability by redundancy," *Inform. and Contr.* vol. 1, pp. 127–147; May, 1958.

[583] Lowenschuss, O. "Non-binary switching theory," 1958 IRE National Convention Record, vol. 6, pt. 4, pp. 305–317.

[584] Lupanov, O. B., "O sinteze kontaktnykh skhem" (On the synthesis of contact networks), *Dokl. AN SSSR*, vol. 119, no. 1, pp. 23–26; March, 1958. (In Russian.) *Automation Express*, vol. 1, pp. 7–8, September, 1958. (Transl.) Dissertation, V. A. Steklov Mat. Inst., 5 pp.; 1958. (In Russian.)

[585] Lupanov, O. B., "Ob odnom metode sinteza skhem" (On a special circuit synthesis method), *Izb. Byssh. Uchebn. Zavedenii. Radiofiz*, vol. 1, p. 120; 1958. (In Russian.)

[586] Lupanov, O. B., "Ob odnom probleme Shennona" (On a certain Shannon problem), *Uspekhi Mat. Nauk*, vol. 13, no. 4, p. 211; July–August, 1958. (In Russian.) *Automation Express*, vol. 1, no. 5, p. 11; January, 1959. (Excerpt.)

[587] Markov, A. A., "On the inversion complexity of a system of functions," *J. Assoc. Comput. Mach.*, vol. 5, no. 4, pp. 331–334; 1958.

[588] McCluskey, E. J., Jr., "Iterative combinational switching networks—general design considerations," IRE Trans. on Electronic Computers, vol. EC-7, pp. 285–291; December, 1958.

[589] Mekler, Ja. I., "Uproshchennyi algebraicheskii metod sinteza relejnykh skhem" (Simplified algebraic synthesis of relay circuits), *Avtomat. i Telemekh.*, vol. 19, no. 12, pp. 1129–1144; 1958. (In Russian.) *Automation and Remote Control*, vol. 19, no. 12, pp. 1099–1114; 1958. (Transl.)

[590] Miller, R. E., "Formal analysis and synthesis of bilateral switching networks," IRE Trans. on Electronic Computers, vol. EC-7, pp. 231–244; September, 1958.

[591] Moisil, Gr. C., "Zarys algebry automatycznych ukladow przekaznikowo stykowych" (Sketch for an algebra method for automatic determining contact transmittance), *Zastowania Mat.*, vol. 4, no. 1, pp. 1–27; 1958. (In Polish.)

[592] Moisil, Gr. C., "Intrebuintarea logicilor trivalente in teoria mecanismelor automate. V. Schemele P-1" (Application of three-valued symbolic logic to the theory of automatic devices. V. Circuits P-1), *Commun. Acad. rep. populare Romîne*, vol. 8, p. 1127; 1958. (In Rumanian.)

[593] Moisil, Gr. C., "Sur un theoreme d'existence dans la théorie algebrique des mecanismes automatiques discrets" (On an existence theorem in the algebraic theory of automatic devices), *Rev. math. pur. appl.*, vol. 3, no. 1, pp. 1–15; 1958. (In French.)

[594] Moisil, Gr. C., "Teoria algebrica dei mecanismi automatici" (Algebraic theory of automatic devices), *Conf. Seminario Mat. Univ. Bari*, pp. 33–34; 1958. (In Rumanian.)

[595] Moore, E. F., "A Table for Four-Relay Two-Terminal Contact Networks," Bell. Lab. Tech. Memo. MM-52-180-45, case 22108.

[596] Moore, E. F., "Table of four-relay contact networks," in "The Design of Electric Circuits," by R. Higonnet and R. Gréa, McGraw-Hill Book Co., Inc., New York, N. Y., Appendix III; 1958.

[597] Moskowitz, F., "The analysis of redundancy networks," *Trans. AIEE (Commun. and Electronics)*, vol. 77, no. 39, pp. 627–632; November, 1958.

[598] Mühldorf, E., "Ternäre Schaltalgebra" (Ternary switching-circuit algebra), *Arch. elektrischen. Übertr.*, vol. 12, no. 3, pp. 138–148; March, 1958. (In German.)

[599] Mühldorf, E., "Schaltungen für ternäre Schaltvariable" (Circuits for ternary switching variables), *Arch. elektrischen. Übertr.*, vol. 12, no. 4, pp. 176–182; April, 1958. (In German.)

[600] Muller, D. E., "Asynchronous switching theory," Session Advanced Theory Logic Design Digital Comput., University of Michigan, Ann Arbor; June, 1958.

[601] Mullin, A. A., "Reliable stochastic sequential switching circuits," *Trans. AIEE (Commun. and Electronics*, no. 39), vol. 77, pp. 606–611; November, 1958.

[602] Mullin, A. A., and W. Kellner, "A residue test for Boolean functions," *Trans. Ill. State Acad. Sci.*, vol. 51, no. 3–4, pp. 14–19; 1958.

[603] Nakagawa, N., "On evaluation of the graph trees and the driving-point admittance," IRE Trans. on Circuit Theory, vol. CT-5, pp. 122–127; June, 1958.

[604] Naslin, P., "Introduction à l'étude des automatismes à séquences" (Introduction to the study of sequential networks), *Automatisme*, vol. 3, no. 1, pp. 3–11; no. 2, pp. 43–49; no. 3, pp. 87–94; no. 4, pp. 135–143; no. 5, pp. 179–185; no. 6, pp. 225–228; no. 7, pp. 260–264; 1958. (In French.)

[605] Nechiporuk, E. I., "O sinteze skhem s pomoshchia lineijnykh preobrazovanij peremennykh" (On the synthesis of networks using linear transformations of the variables), *Dokl. AN SSSR*, vol. 123, no. 4, pp. 610–612; December, 1958. (In Russian.) *Automation Express*, vol. 1, no. 8, pp. 12–13; April, 1959. (Transl.)

[606] Nerode, A., "Linear automata transformations," *Proc. Am. Math. Soc.*, vol. 9, pp. 541–544; 1958.

[607] Netherwood, D. B., "Matrix Representation of Boolean Forms," 5th Ann. Symp. Comput. Data Process., Denver Research Inst., University of Denver, Colo.; July, 1958.

[608] Neumann, P., "A Markov Model for Sequential Processes," Bell Labs. Inc., Rept. No. 19, pp. III 1–III 56; February, 1958.

[609] Ninomiya, I., "A theory of the coordinate representation of switching functions," *Mem. Fac. Engrg. Nagoya Univ.*, vol. 10, no. 2, p. 175–190; November, 1958.

[610] Oifa, I. A., "Inversirovanie 'neplanarnykh' skhem" (Inversion of nonplanar circuits), *Dokl. AN SSSR*, vol. 118, no. 5, pp. 928–929; 1958. (In Russian.) *Automation Express*, vol. 1, no. 2, p. 16; July, 1958. (Excerpt.)

[611] Ostianu, V. M., "Problemy teorii ustrojstv relejnogo dejstvija" (Problems of the theory of relay devices), *Vestnik AN SSSR*, vol. 1, pp. 131–132; 1958. (In Russian.)

[612] Poletaev, I. A., "Teoreticheskaja logika i algebra relejnykh skhem" (Theoretical logic and the algebra of relay circuits), in "Signals, Izdatel'stvo 'Sovetskoe Radio'," p. 404; 1958. (In Russian.)

[613] Portela, A. G., "Aplicacao da logica simbolica aos sistemas eléctricos" (Application of symbolic logic in electric systems), *Técnica, (Lisbon)*, no. 278, pp. 253–264; January, 1958. (In Portuguese.)

[614] Povarov, G. N., "O gruppovoij invariantnosti Bulevykh funktsij" (On group invariance of Boolean functions), *Analele stiint. Univ. din Iasi*, vol. 4, Sect. 1, pp. 39–41; 1958. (In Russian.)

[615] Povarov, G. N., and V. N. Roginskij, "Grafichnij metod sintezu kontaktnykh bogatpoljusnikov" (Graphical method for synthesizing multiterminal contact networks), *Avtomatyka, (Kiev)*, vol. 3, pp. 84–91; 1958. (In Russian.)

[616] Pugmire, G. M., and A. Rose, "Formulae, corresponding to universal decision elements," *Z. math. Logik Gr. Math.* vol. 4, p. 9; 1958.

[617] Raney, G. N., "Sequential functions," *J. Assoc. Comput. Mach.*, vol. 5, pp. 177–180; April, 1958.

[618] Reza, F., "Some topological considerations in network theory," IRE Trans. on Circuit Theory, vol. CT-5, pp. 30–42; March, 1958.

[619] Rieger, L., "O teorii neuronových siti" (Theory of indeterminate circuits), *Aplikace Mat.*, vol. 3, no. 4, pp. 243–274; 1958. (In Czech.)

[620] Righi, R., "L'algebra di commutazione dei circuiti electrici con elementi commutatori a piu posizioni" (Algebra of switching circuits with multipositional switching elements), *Ing. ferroviaria*, vol. 13, no. 10, pp. 881–898; vol. 13, no. 11, pp. 1019–1030; October–November, 1958. (In Italian.)

[621] Roginskij, V. N., "Mashina dlja postroenija relejnykh skhem" (Machine for synthesizing relay networks), *Vestnik AN SSSR*, no. 10, pp. 71–73; October, 1958. (In Russian.) *Automation Express*, vol. 1, no. 9, pp. 5–6; May, 1959. (Excerpt.)

[622] Rose, A., "Many-valued logical machines," *Proc. Cambridge Phil. Soc.*, vol. 54, pt. 3, pp. 307–321; 1958.

[623] Rose, A., "The use of universal decision elements as flip-flops," *Z. math. Logik Gr. Math.*, vol. 4, pp. 169–174; 1958.

[624] Roth, J. P., "Algebraic topological methods for the synthesis of switching systems," *Trans. Am. Math. Soc.*, vol. 88, pp. 301–326; July, 1958.

[625] Rouche, N., "Some properties of Boolean equations," IRE Trans. on Electronic Computers, vol. EC-7, no. 4, pp. 291–298; December, 1958.

[626] Rudeanu, S., "Intrebuintarea imaginarelor lui Galois in teoria mecanismelor automata. X. Clasificarea functiilor de doua variable in GF [22]," (Application of Galoisian fields in the theory of automatic devices. X. Classification of the functions of two variables in GF [22]), *Studii si Cercetari mat.*, no. 1, Bucuresti; 1958. (In Rumanian.)

[627] Sageau, A., "Les circuits logiques. Principes et applications" (Logical circuits. Principles and applications), *Docaéro*, no. 51, pp. 57–60; 1958. (In French.)

[628] Sawicki, Z., "The transformer realization of proposition functions," *Bull. acad. polon. sci.*, Ser. sci. tech., vol. 6, no. 2, pp. 111–116; 1958.

[629] Schubert, E. J., "Matrix analysis of logical networks," *Trans. AIEE (Commun. and Electronics)*, vol. 77, no. 35, pp. 10–13; March, 1958.
[630] Semon, W., "Synthesis of series-parallel network switching functions," *Bell Sys. Tech. J.*, vol. 37, no. 4, pp. 877–898; July, 1958.
[631] Seshu, S., "Theory of linear graphs with applications in electrical engineering," Syracuse University, Syracuse, N. Y., 1958.
[632] Shannon, C. E., "Von Neumann's contribution to automata theory," *Bull. Am. Math. Soc.*, vol. 64, no. 3, pt. 2, pp. 123–129; 1958.
[633] Simon, J. M., "Some aspects of the network analysis of sequence transducers," *J. Franklin Inst.*, vol. 65, pp. 439–450; June, 1958.
[634] Smirnov, A. S., "Sintez i analiz elektronno-lampovykh perekljuchajushchikh [codnotaktnykh] skhem" (Synthesis and analysis of vacuum-tube switching [nonsequential] circuits). *Trudy AN Krylov*, vol. 16, pp. 65–82; 1958. (In Russian.)
[635] Smirnov, A. S., "O kolichestve tipov Bulevykh funktsij i kolichestve Bulevykh funktsij odnogo i togo zhe tipa" (On the number of types of Boolean functions and the number of Boolean functions of one and the same type), *Trudy AN Krylov*, vol. 16, pp. 105–123; 1958. (In Russian.)
[636] Smith, J. J., "Relay circuit analysis—a new approach," *Nuclear Engrg.*, vol. 2, no. 22, pp. 5–8; 1958.
[637] Soubies-Camy, H., "L'algèbre logique appliquée aux techniques binaires" (Logical algebra applied to binary techniques), *Automatisme*, vol. 3, no. 7, pp. 266–272; no. 9, pp. 338–345; no. 10, pp. 385–393; no. 11, pp. 423–429; 1958. (In French.)
[638] Srinivasan, C. V., and R. Narasimhan, "On the Synthesis of Finite Sequential Machines," Tata Inst. Fundamental Res., Bombay; 1958.
[639] Svoboda, A., "Navrh hradlovych obvodu prokusnoh cestou" (Graphical aids to minimization in switching circuits), *Stroje na zpracovani informaci, Sbornik VI*, Prague, pp. 35–53; 1958. (In Czech.)
[640] Tompkins, H. E., and R. McNaughton, *et al.* "General Switching Theory," Moore School Engrg., University of Pennsylvania, Philadelphia, Rept. No. 59-02, 1st Quart. Progress Rept.; October, 1958.
[641] Trakhtenbrot, B. A., "K teorii bespovtornykh kontaktnykh skhem" (On the theory on nonrepetitive contact networks), in (Collected papers on mathematical logic and its application to certain questions of cybernetics), *Izd. AN SSSR*, pp. 226–269; 1958. (In Russian.) *Trudy V. A. Steklov Mat. Inst.*, (51).
[642] Trakhtenbrot, B. A., "Sintez logicheskykh tsepei operatory kotorykh opisyvajutsia sredstvami ischislenia odnomestnykh predikatov" (Synthesis of logical nets whose operators are describable in terms of one-place predicate calculus), *Dokl. AN SSSR*, vol. 118, no. 4, pp. 646–648; 1958. (In Russian.)
[643] Tsetlin, M. L., "O kompozitsii i rasbienjakh neprimitivnykh skhem" (On combinations and subdivisions of nonprimitive networks), *Dokl. AN SSSR*, vol. 118, no. 3, pp. 488–491; January, 1958. (In Russian.) *Automation Express*, vol. 1, pp. 11–13; July, 1958. (Transl.)
[644] Tsetlin, M. L., "O neprimitivnykh skhemakh" (On nonprimitive networks), *Problems Cybernetics, Fitzmatgiz, (Moscow)*, vol. 1, pp. 23–45; 1958. (In Russian.)
[645] Tsetlin, M. L., and G. S. Eidus, "Matrichnykh metod sinteza skhem svjazi i upravlenija" (Matrix method for synthesizing multicontact relay networks in communication and control), *Elektrosvjaz'*, no. 4, pp. 41–48; April, 1958. (In Russian.) *Automation Express*, vol. 1, pp. 13–16; July, 1958. (Excerpt.)
[646] Valle, A. G., del, "Filosofia de las redes. Principio fundamentales" (Philosophy of schemes. Fundamental principles), *Calc. auto. y. cib.*, no. 19, pp. 16–20; 1958. (In Spanish.)
[647] van der Haer, F. W., (The composition of the system of two-terminal networks formed by the contacts of n switching units), *P.T.T.-Bedrijf* vol. 8, no. 3, pp. 154–163; July, 1958.
[648] Voishvillo, E. K., "Metod uproshchenija form vyrazhenija funktsij istinnosti" (Method for simplifying the form of truth functions), *Nauch. Dokl. Vyssh. Skholy, Filosof. Nauki 2*, pp. 120–135; 1958. (In Russian.)
[649] Wallace, J. H., "Mechanical method for designing switching circuits, *Western Elec. Engr.*, vol. 11, no. 4, pp. 43–48; October, 1958.
[650] Warfield, J. N., "A note on the reduction of switching functions," IRE Trans. on Electronic Computers, vol. EC-7, pp. 180–181; June, 1958.
[651] Warfield, J. N., "Switching circuits as topological models in discrete probability theory," IRE Trans. on Electronic Computers, vol. EC-7, pp. 251–252; September, 1958.
[652] Weeg, G. P., "Folding of symmetric functions," IRE Trans. on Electronic Computers, vol. EC-7, p. 325; December, 1958.
[653] Weinberg, L., "Number of trees in a graph," Proc. IRE, vol. 46, pp. 1954–1955; December, 1958.
[654] Weitzsch, F., "Von der Aussagelogik zur logischen Algebra" (From symbolic logic to logical algebra), *Elektron. Rundschau.*, vol. 12, no. 5, pp. 160–164; May, 1958. (In German.)
[655] Wolpe, H., "Algorithm for analyzing logical statements to produce a truth function table," *Commun. Assoc. Comput. Mach.*, vol. 1, no. 3, pp. 4–12; March, 1958.
[656] Zemanek, H., "Die Lösung von Gleichungen in der Schaltalgebra" (Solving the equations of the switching circuit algebra), *Arch. elekt. Übertr.*, vol. 12, no. 1, pp. 35–44; January, 1958. (In German.)
[657] Zhuravlev, Ju. I., "Ob optimal'nykh algoritmakh vybora" (On optimal-choice algorithms), *Dokl. AN SSSR*, vol. 121, no. 13, pp. 411–414, 1958. (In Russian.)
[658] Zhuravlev, Ju. I., "Ob otdelimosti podmnozhestv vershin n-mernogo edinichnogo kuba" (On separability of subsets of the vertices of an n-dimensional unit cube), *Dokl. AN SSSR*, vol. 113, no. 2, pp. 264–267; 1957. (In Russian.) *Trudy Inst. Mat.*, vol. 51; 1958. (In Russian.)

Books:

[659] Bazilevskij, "Doklady na konferentsii po teorii i primeneniju diskretnykh avtomaticheskikh sistem" (Conference on the theory and applications of discrete automatic systems), Inst. Avtomat. i Telemekh., AN SSSR, Moscow; 1958. (In Russian.)
[660] Caldwell, S. H., "Switching Circuits and Logical Design," John Wiley and Sons, Inc., New York, N. Y., 686 pp., 1958; second printing, 1959. Proc. IRE, vol. 46, p. 1887; November, 1958. (Review by R. W. Stuart.)
[661] Chegis, *et al.*, "Sbornik st. po mat. logike i ee prilozhenija k nekotorym voprosam kibernetiki" (Collected papers on mathematical logic and its applications to certain questions of cybernetics), Izd. AN SSSR, Moscow; 1958. (In Russian.)
[662] Glushov, "Tezisy dokl. soveshchanija po vychislitel'noj mat. i primeneniju sredstv vychislitel'noj tekhnika" (Conference on computing mathematics and the applications of computer technology), Izd. AN Azerb SSR, Baku; 1958. (In Russian.)
[663] Higonnet, R. A., and R. A. Gréa, "Logical Design of Electrical Circuits," McGraw-Hill Book Co., Inc., New York, N. Y., 220 pp.; 1958.
[664] Humphrey, W. S., Jr., "Switching Circuits with Computer Applications," McGraw-Hill Book Co., Inc., New York, N. Y., 264 pp.; 1958. Proc. IRE, vol. 47, p. 346; February, 1959. (Review by M. Karnaugh.)
[665] Naslin, P., "Circuits a relais et automatismes à sequences" (Relay circuits and sequential networks), Dunod et Cie, Paris, 229 pp.; 1958. (In French.)
[666] Riordan, J., "An Introduction to Combinatorial Analysis," John Wiley and Sons, Inc., New York, N. Y., 244 pp.; 1958. *J. Assoc. Comput. Mach.*, vol. 5, p. 389; October, 1958. (Review by M. Newman.) (See also [572], [595], and [612].)

Dissertations:

[667] Atrubin, A. J., "A Study of Several Planar Iterative Switching Circuits," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 98 pp.; 1958.
[668] Brandt, P. W., "A Circuit for Minimizing Boolean Functions," Elec. Engrg. Dept., Mass. Inst. Tech., Cambridge, 34 pp.; 1958.
[669] Hennie, F. C., "Analysis of One-Dimensional Iterative Logical Circuits," Mass. Inst. Tech., Cambridge, 104 pp.; 1958.
[670] Mekler, Ja. I., "Uproshchennye metody sinteza relejnykh skhem i perekhodnye rezhimij" (Simplified methods of synthesizing relay circuits and transient response), Inst. Avtomat. i Telemekh., AN SSSR, Moscow, 20 pp.; 1958. (In Russian.)
[671] Tsetlin, M. L., "Matrichnykh metod sinteza relejnykh skhem i nekotorye eg o prilozhenija" (Matrix method for synthesizing relay circuits and certain of its applications), V. A. Steklov Mat. Inst., Moscow, 8 pp.; 1958. (In Russian.)

Bibliographical References

[A-1] Belevitch, V., "Recent Russian publications on switching theory," IRE Trans. on Circuit Theory, vol. CT-3, pp. 77–79; March, 1956.
[A-2] Gavrilov, M. A., "Sovremennoe sostojanie i osnovnye problemy issledovatel'skoj raboty v oblasti teleupravlenija" (The present state of the research on the basic problems of telecontrol), in "Telemekh. v narodnom khozjajstve" (Telemechanization in the National Economy), Acad. Sci. USSR Press, Moscow, pp. 11–50; 1956. (In Russian.)
[A-3] Gavrilov, M. A., "A survey of research in the theory of relay networks in the USSR," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 1, pp. 26–53; April, 1957.

[A-4] Gavrilov, M. A., and G. A. Shastova, "Osnovnye voprosy teorii postroenija signalov i teorii pomekhoustojchivosti v ustrojstvakh teleupravlenija" (Fundamental questions of the theory of designing coding systems and of the theory of noise-eliminating devices in remote-control circuits), *Trudy Sessii Obshchego Sobranija Akad. Nauk SSSR* (*Proc. General Assembly Acad. Sci. USSR*), Moscow; 1957. (In Russian.)

[A-5] Goncharov, V. P., "Seminar po tekhnicheskim prilozhenijam mathematicheskoj logiki [1958–1959 gg.]" (Seminar on the technical applications of mathematical logic [the years 1958–1959]), *Avtomat. i Telemekh.*, vol. 21, no. 1, pp. 145–148; January, 1960. (In Russian.) *Automation and Remote Control*, vol. 21, no. 1, pp. 98–100; January, 1960. (Transl.)

[A-6] Grebenshchikov, V. N., "Seminar po tekhnicheskim prilozhenijam matematicheskoj logiki [1957–1958 gg.]," (Seminar on the technical applications of mathematical logic [the years 1957–1958)], *Avtomat. i Telemekh.*, vol. 20, no. 1, pp. 97–99; January, 1959. (In Russian.) *Automation and Remote Control*, vol. 20, no. 1, pp. 91–94; January, 1959. (Transl.)

[A-7] Kazakov, V. D., and O. P. Kuznetsov, "Spisok inostrannykh rabot po teorii ustroistv relejnogo dejstvija i konechnykh avtomatov za 1958 g." (List of foreign literature on the theory of relay devices and finite automata for the year 1958), *Avtomat. i Telemekh.*, vol. 21, no. 9, pp. 1332–1338; September, 1960. (In Russian.) *Automation and Remote Control*, vol. 21, no. 9, September, 1960. (Transl.)

[A-8] Koval, N. S., "Seminar po tekhnicheskim prilozhenijam matematicheskoj logiki [1955–1957 gg.]" (Seminar on the technical applications of mathematical logic [The years 1955–1957]), *Avtomat. i Telemekh.*, vol. 18, no. 10, pp. 950–952; October, 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 10; October, 1957. (Transl.) *Automation Express*, vol. 1, pp. 3–4; May, 1958. (Excerpt.)

[A-9] Moskatov, G. K., "Spisok inostrannoj literatury po teorii ustrojstv relejnogo dejstvija za 1956 g.," (List of foreign literature on the theory of relay devices for the year 1956), *Avtomat. i Telemekh.*, vol. 19, no. 10, pp. 992–995; October, 1958. (In Russian.) *Automation and Remote Control*, vol. 19, no. 10, pp. 971–974; October, 1958. (Transl.)

[A-10] Moskatov, G. K., "Spisok otechestvennoj i perevodnoj literatury po teorii ustrojstv relejnogo dejstvija za 1957 g." (List of domestic and foreign literature on the theory of relay devices for the year 1957), *Avtomat. i Telemekh.*, vol. 19, no. 12, pp. 1150–1154; December, 1958. (In Russian). *Automation and Remote Control*, vol. 19, no. 12, pp. 1121–1125; December, 1958. (Transl.) *Automatation Express*, vol. 1, no. 7, pp. 6–8; March, 1959. (Transl.)

[A-11] Moskatov, G. K., "Spisok inostrannoj literatury po teorii ustrojstv relejnogo dejstvija za 1957 g." (List of foreign literature on the theory of relay devices for the year 1957), *Avtomat. i Telemekh.*, vol. 20, no. 2, pp. 267–269; February, 1959. (In Russian.) *Automation and Remote Control*, vol. 20, no. 2, pp. 256–258; February, 1959. (Transl.)

[A-12] Moskatov, G. K., "Spisok otechestvennykh rabot po teorii relejnykh skhem i konechnykh avtomatov za 1958 g." (List of domestic works on relay circuit theory and finite automata for the year 1958), *Avtomat. i Telemekh.*, vol. 20, no. 8, pp. 1148–1150, August, 1959. (In Russian.) *Automation and Remote Control*, vol. 20, no. 8, pp. 1116–1119; August, 1959. (Transl.)

[A-13] Netherwood, D. B., "Logical machine design: a selected bibliography," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-7, pp. 155–178; June, 1958.

[A-14] Netherwood, D. B., "Logical machine design II: a selected bibliography," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-8, pp. 367–380; September, 1958.

[A-15] Piesch, J., and H. Sequenz, "Österreichische Wegbereiter der Theorie elektrischer Schaltungen" (Austrian pioneers of the theory of switching circuits), *Elektrotech. u. Maschinenbau*, vol. 75, no. 9, pp. 241–245; 1958. (In German.)

[A-16] Povarov, G. N., "Spisok inostrannoj i perevodnoj literatury po teorii relejno-kontaktnykh skhem za 1950–1954 gg." (List of foreign and domestic literature on the theory of relay contact networks for the years 1950–1954), *Avtomat. i Telemekh.*, vol. 16, no. 4, pp. 412–420; 1955. (In Russian.)

[A-17] Povarov, G. N., "Spisok otechestvennoj i perevodnoj literatury po teorii relejnykh skhem za 1956 g." (List of domestic and foreign literature on the theory of relay circuits for the year 1956), *Avtomat. i Telemekh.*, vol. 18, no. 12, pp. 1151–1152; December, 1957. (In Russian.) *Automation and Remote Control*, vol. 18, no. 12, pp. 1196–1197; December, 1957. (Transl.)

[A-18] Santesmases, J. G., "Switching research in Spain," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, Mass., pt. 2, pp. 99–114; April, 1957.

[A-19] Stumpers, F. L., "A bibliography of information theory, communication theory cybernetics," Research Lab. of Electronics, Mass. Inst. Tech., Cambridge, dissertation, 49 pp., February, 1953; IRE TRANS. ON INFORMATION THEORY, vol. IT-2, November, 1953; vol. IT-1, pp. 31–47, September, 1955; vol. IT-3, pp. 150–166, June, 1957; vol. IT-6, pp. 25–51, March, 1960.

[A-20] Walther, A., "Switching research in Germany," *Proc. Internatl. Symp. Theory of Switching*, Harvard University, Cambridge, pt. 2, pp. 295–301; April, 1957.

BOOKS

[B-1] Aiken, H. H., and Staff of Comput. Lab. Harvard University, "Synthesis of electronic computing and control circuits," *Ann. Comput. Lab. Harvard Univ.*, Cambridge, Mass., vol. 27, 278 pp.; 1951.

[B-2] Ashby, W. R., "An Introduction to Cybernetics," Chapman [397] and Hall, London; John Wiley and Sons, Inc., New York, N. Y., 295 pp.; 1956.

[B-3] Bazilevskij, "Doklady na konferentsii po teorii i primeneniju [659] diskretnykh avtomaticheskikh sistem" (Proc. of conference on the theory and applications of discrete automatic systems), Institut Avtomat. i Telemekh., AN SSSR, Moscow; 1958. (In Russian.)

[B-4] Berkeley, E. C., "A Summary of Symbolic Logic and Its [256] Practical Applications," E. C. Berkeley and Assoc., New York, N. Y.; June, 1954.

[B-5] Berkeley, E. C., "Circuit Algebra—Introduction," E. C. [312] Berkeley and Assoc., New York, N. Y.; August, 1955.

[B-6] Caldwell, S. H., "Switching Circuits and Logical Design," [660] John Wiley and Sons, Inc., New York, N. Y., 686 pp.; 1958.

[B-7] Chegis, *et al.*, "Sbornik st. po matem. logike i ee prilozhenija k [661] nekotorykh voprosam kibernetiki" (Collected papers on mathematical logic and its applications to certain questions of cybernetics), Izd. AN SSSR, Moscow; 1958. (In Russian.)

[B-8] *Proc. 3rd Eastern Joint Computer Conf.*, Washington, D. C., December 8–10, 1953, IRE, New York, N. Y., 125 pp.; 1954. (Theme: Information Processing Systems—Reliability and Requirements.)

[B-9] *Proc. 4th Eastern Joint Computer Conf.*, Philadelphia, Pa., December 8–10, 1954, AIEE, New York, N. Y., 92 pp.; 1956. (Theme: Design and Application of Small Digital Computers.)

[B-10] *Proc. 5th Eastern Joint Computer Conf.*, Boston, Mass., November 7–9, 1955, IRE, New York, N. Y., 100 pp.; 1956. (Theme: Computers in Business and Industrial Systems.)

[B-11] *Proc. 6th Eastern Joint Computer Conf.*, New York, N. Y., December 10–12, 1956, AIEE, New York, N. Y., 150 pp.; 1957. (Theme: New Developments in Computers.)

[B-12] *Proc. 8th Eastern Joint Computer Conf.*, Philadelphia, Pa., December 3–5, 1958, AIEE, New York, N. Y., 184 pp.; 1959. (Theme: Modern Computers: Objectives, designs, applications.)

[B-13] Edler, R., "Entwurf von Schaltungen und Schaltapparaten [26] [Schaltungstheorie]" (Outline of Contact Networks and Switching Devices [Switching Theory]), Dr. Max Jänecke Verlagsbuchhandel, Hanover; 1905. (In German.) "Schaltlehre [Wege zum Schaltplan]" (Switching Theory [Ways to the Switching System]), Dr. Max Jänecke Verlagsbuchhandle, Leipzig, 2nd ed.; 1927. (In German.)

[150] "Schaltlehre—Entwicklung und Anwendungen" (Switching Theory—Development and Applications), Franz Deuticke Verlag, Wien, 3rd ed.; 1952. (In German.)

[B-14] "Elektronischen Rechenmaschinen und Informationsverarbeit-[398] ung" (Electronic Digital Computers and Data Processing), Nachrtech. Fachberichte, Verlag Friedrich Vieweg and Sohn, Braunschweig, vol. 4, 237, pp.; 1956. (In German.)

[B-15] Gavrilov, M. A., "Teorija relejno-kontaktnykh skhem" [101] (Theory of relay contact networks), Izd. AN SSSR, Moscow, 305 pp.; 1950. (In Russian.)

[188] "Relaisschalttechnik für Stark- und Schwachstromsanlagen" (Relay Switching Networks for Electrical and Electronic System), VEB Verlag Technic, Berlin, 324 pp.; 1953. (German transl.)

[B-16] Gavrilov, M. A., V. A. Malov, V. A. Zhozhikashvili, *et al.*, [399] Eds., "Telemekhanizatsija v narodnom khozjajstve" (Telemechanization in the National Economy), Izd. AN SSSR, Inst. Avtomat. i Telemekh., Moscow, 481 pp.; 1956. (In Russian.)

[B-17] Glushov, Ed., "Tezisy doklady soveshchanija po vychis-
[662] litel'noj mat. i primeneniju sredstv vychislitel'noj tekhnika" (Conference on Computing Mathematics and the Application of Computer Techniques), Izd. AN Azerb. SSR, Baku; 1958. (In Russian.)

[B-18] Higonnet, R. A., and R. A. Gréa, "Étude Logique des Circuits
[313] Electriques et des Systèmes Binaires" (Logical Study of Electrical Circuits and Binary Systems), Berger-Levrault, Paris, 425 pp.; 1955. (In French.)

[B-19] Higonnet, R. A., and R. A. Gréa, "Logical Design of Elec-
[663] trical Circuits," McGraw Hill Book Co., Inc., New York, N. Y., 220 pp.; 1958. (See Appendix III by E. F. Moore, "Table of Four-Relay Contact Networks.")

[B-20] Humphrey, W. S., Jr., "Switching Circuits with Computer
[664] Applications," McGraw-Hill Book Co., Inc., New York, N. Y., 264 pp.; 1958.

[B-21] Karmazov, M. G., "Avtomaticheskaja Telefonija" (Auto-
[189] matic Telephony), Svjaz'izdat Press, Moscow 3rd. ed., 290 pp.; 1953. (In Russian.)

[B-22] Keister, W., A. E. Ritchie, and S. H. Washburn, "The De-
[116] sign of Switching Circuits," D. Van Nostrand Co., Inc., New York, N. Y., 556 pp.; 1951.

[B-23] Lewis, C. I., and C. H. Langford, "Symbolic Logic," Dover Press, New York, N. Y.; 1932.

[B-24] Lischke, R., "Schaltlehre—Anleitung zur Ansmittlung von
[27] Schaltungen elektrischer Einrichtungen" (Switching Theory—Guidance in Development of Switching Networks in Electrical Systems), Verlag Hachmeister und Thal, Leipzig; 1911. 3rd. ed., 1921. (In German.)

[B-25] Markhaj, E. V., and I. A. Babitskij, "Avtomaticheskaja tele-
[102] fonija" (Automatic Telephony), Svjaz'izdat, Moscow; 1950. (In Russian.)

[B-26] Markov, A. A. (Theory of Algorithms), Izd. AN SSSR,
[257] Moscow-Leningrad; 1954. (Original in Russian.)

[B-27] Naslin, P., "Circuits á Relais et Automatismes à Sequences"
[665] (Relay Circuits and Sequential Switching Networks), Dunod, et Cie, Paris, 229 pp.; 1958. (In French.)

[B-28] Packard, C. A., "Relay Engineering," Struthers-Dunn, Inc.,
[64] Philadelphia, Pa., 640 pp.; 1946.

[B-29] Plechl, O., "Elektromechanische Schaltungen und Schalt-
[400] geräte" (Electromechanical Contact Networks and Switching Devices), Springer-Verlag, Wien, 224 pp.; 1956. (In German.)

[B-30] Riordan, J., "An Introduction to Combinatorial Analysis,"
[666] John Wiley and Sons, Inc., New York, N. Y., 244 pp.; 1958.

[B-31] Roginskij, V. N., and A. D. Kharkevich, "Relejnye skhemy v
[314] telefonii" (Relay Circuits in Telephony), Tipografija Svjaz'izdata, Moscow, 166 pp.; 1955. (In Russian.)

[B-32] Shannon, C. E., and J. McCarthy, Eds., "Automata Studies,"
[401] Princeton University Press, Princeton, N. J., 285 pp.; 1956.

[B-33] Shestakova, V. I., Transl. Ed., "Sintez elektronnykh vychis-
[258] litel'nykh i upravljajushchikh skhem" (Synthesis of electronic computing and control circuits), Izd. Inostrannoj Literatury, Moscow; 1954. (In Russian.)

[B-34] "Tezisy dokladov na vsesojuznom soveshchanii po teorii re-
[525] lejnykh skhem" (Thesis of the papers read at the All-Union Moscow Conf. on the Theory of Relay Operating Devices), Izd. AN SSSR, Moscow; 1957. (In Russian.)

[B-35] *Proc. Western Computer Conf.*, Los Angeles, Calif., February 4–6, 1953, IRE, New York, N. Y., 231 pp.; 1953.

[B-36] *Proc. Western Computer Conf.*, Los Angeles, Calif., February 11–12, 1954, AIEE, New York, N. Y., 191 pp.; 1954.

[B-37] *Proc. Western Joint Computer Conf.*, Los Angeles, Calif., March 1–3, 1955, IRE, New York, N. Y., 132 pp.; 1955.

[B-38] *Proc. Western Joint Computer Conf.*, San Francisco, Calif., February 7–9, 1956, AIEE, New York, N. Y., 142 pp.; 1956.

[B-39] *Proc. Western Joint Computer Conf.*, Los Angeles, Calif., February 26–28, 1957, IRE, New York, N. Y., 240 pp.; 1957.

[B-40] *Proc. Western Joint Computer Conf.*, Los Angeles, Calif., May 6–8, 1958, AIEE, New York, N. Y., 244 pp.; 1958.

[B-41] *Proc. Western Joint Computer Conf.*, San Francisco, Calif., March 3–5, 1959, IRE, New York, N. Y., 360 pp.; 1959.

[B-42] Weyh, U., "Elemente der Schaltungsalgebra" (Elements of the Algebra of Switching Circuits), R. Oldenbourg Verlag, München, 116 pp.; 1960. (In German.)

Author Index

Popovici, C.: 1956 [364], [369]
Portela, A. G.: 1958 [613]
Porter, A.: 1958 [546]
Postley, J. A.: 1955 [294]
Pot, J. J.: 1950 [99]
Povarov, G. N.: 1954 [231]–[233], [261]; 1955 [295]–[298]; 1956 [370]–[376]; 1957 [485]–[488]; 1958 [614], [615], [A-16], [A-17]
Prager, E.: 1951 [113]
Prim, R. C.: 1957 [489]
Pugmire, G. M.: 1958 [616]
Puig-Adam, P.: 1952 [140]

Quine, W. V.: 1936 [23]; 1945 [57]; 1952 [141]; 1953 [173]; 1955 [299]

Ramlau, P.: 1946 [62]; 1948 [78]
Raney, G. N.: 1958 [617]
Raspanti, M.: 1953 [174]
Rettig, A. S.: 1953 [156]
Reza, F.: 1958 [618]
Rieger, L.: 1958 [619]
Righi, R.: 1954 [234]–[236]; 1955 [300]; 1956 [377]; 1958 [620]
Riguet, J.: 1953 [175]; 1956 [378]
Riordan, J.: 1942 [46]; 1958 [666], [B-30]
Ritchie, A. E.: 1949 [86]; 1951 [116], [B-22]
Ritter, A.: 1938 [34]; 1941 [42]
Rodin, V. N.: 1957 [490]; 1958 [556]
Rodriguez, J. S.: 1956 [387]
Roginskij, V. N.: 1953 [176]; 1954 [237], [238]; 1955 [301], [314], [B-31]; 1956 [379], 1957 [414], [491]–[496]; 1958 [615], [621]
Rohleder, H.: 1954 [239]; 1955 [302]; 1956 [380]; 1957 [497]
Rose, A.: 1958 [616], [622], [623]
Roth, C. H.: 1957 [529]
Roth, J. P.: 1956 [381]–[384]; 1957 [498]; 1958 [624]
Rotolo, L. S.: 1958 [564]
Rouche, N.: 1956 [385]; 1958 [625]
Rozenberg, V.: 1939 [38]; 1940 [41]
Rubinoff, M.: 1957 [499]
Rudeanu, S.: 1958 [626]
Ryser, H. J.: 1957 [500]

Sacerdote, G.: 1956 [386]
Sagalovich, I. L.: 1958 [581]
Sageau, A.: 1958 [627]
Samson, E. W.: 1954 [240]; 1955 [303], [304]
Santesmases, J. G.: 1956 [387]; 1957 [A-18]
Santos, I.: 1958 [532]
Sawicki, Z.: 1958 [628]
Schaefer, D. H.: 1955 [305]
Scheinman, A. H.: 1957 [501]
Schissler, L. R.: 1955 [272]
Schliebs, G.: 1954 [241]
Schmitz, W.: 1952 [142]
Schubert, E. J.: 1958 [629]
Schwab, H.: 1952 [143]
Schwaiger: 1928 [28]
Sebestyen, G.: 1957 [502]
Seidl, L.: 1958 [577]
Sequenz, H.: 1958 [A-15]
Semon, W.: 1952 [144]; 1953 [177]; 1957 [503]; 1958 [630]
Serrell, R.: 1953 [178]
Seshu, S.: 1956 [388]; 1957 [443], [504]; 1958 [631]
Shannon, C. E.: 1938 [33]; 1942 [46]; 1949 [87]; 1953 [179]; 1956 [365]; [401], [B-32]; 1958 [632]
Shastova, G. A.: 1957 [A-4]
Shatsev, N. Z.: 1948 [79]
Shekel, J.: 1953 [180]
Shestakov, V. I.: 1938 [35]; 1941 [43]; 1944 [52]; 1946 [63]; 1953 [181]; 1954 [242]–[245], [258], [B-33]; 1956 [389]; 1957 [505], [506]
Shimbel, A.: 1941 [44]
Shnarevich, D. I.: 1957 [507], [508]
Sikorski, R.: 1956 [390]
Simon, J. M.: 1958 [633]
Singer, T.: 1953 [182]; 1957 [509]
Sittler, R. W.: 1956 [391]
Slepian, D.: 1953 [183]; 1957 [520]
Smirnov, A. S.: 1958 [634], [635]
Smith, J. J.: 1958 [636]
Sobociński, B.: 1953 [184]
Sørvik, R.: 1949 [88]
Soubies-Camy, H.: 1958 [637]
Srinivasan, C. V.: 1958 [638]
Stabler, E. R.: 1944 [53]
Staehler, R. E.: 1952 [145]
Stone, M. H.: 1935 [16]; 1936 [24]
Stumpers, F. L.: 1953 [A-19]
Stutterheim, F. W.: 1952 [123]
Svoboda, A.: 1954 [246]; 1956 [392], [393]; 1957 [510], [511]; 1958 [639]
Svoboda, F.: 1953 [185]; 1954 [512]; 1957 [513], [514]
Szukszta, W.: 1951 [114]

Tellegen, B. D. H.: 1954 [247]
Tezuka, Y.: 1957 [515]
Timoteev, B. L.: 1958 [556]
Tomfel'd, Ju. L.: 1957 [479]
Tompkins, H. E.: 1958 [640]
Torres-Quevedo, L.: 1915 [8]
Touchais, M.: 1953 [186]
Traczyk, T.: 1956 [390]
Trakhtenbrot, B. A.: 1955 [306]; 1956 [394]; 1957 [516]; 1958 [641], [642]
Trent, H. M.: 1954 [248]
Tsetlin, M. L.: 1952 [146]; 1957 [517]; 1958 [643]–[645], [671]
Tsimbalistyj, M.: 1928 [10]
Tsukanov, T. T.: 1956 [409]; 1957 [518]
Tutugan, Fl.: 1950 [100]

Unger, S. H.: 1953 [193]; 1957 [519], [530]
Urbano, R. H.: 1956 [395]
Uyehara, G. U.: 1958 [564]

Vakhnin, M.: 1943 [49]
van der Haer, F. W.: 1958 [647]
van der Poel, W. L.: 1955 [307]
Varnum, E. C.: 1949 [89]; 1951 [115]
Veitch, E. W.: 1952 [147]
Vlodavskij, M.: 1943 [49]
Vogel-Jörgensen, U.: 1954 [249]
Voishvillo, E. K.: 1958 [648]
Volotskij, A.: 1947 [71]
Vonhot: 1952 [148]

Wagner, K. W.: 1952 [149]
Walker, R. M.: 1958 [550]
Wallace, J. H.: 1958 [649]
Walther, A.: 1957 [A-20]
Walzel, O.: 1900 [5]
Warfield, J. N.: 1958 [650], [651]
Washburn, S. H.: 1949 [90]; 1951 [116], [B-22]; 1953 [187]; 1954 [250], [251]
Watanabe, T.: 1954 [252]
Weeg, G. P.: 1958 [652]
Weinberg, L.: 1957 [520]; 1958 [653]
Weinberger, A.: 1957 [521]
Weitzsch, F.: 1955 [308]; 1958 [654]
Wengert, R. E.: 1953 [194]
Weyh, U.: 1960 [B-42]
Whaples, G.: 1949 [81]
Williams, F.: 1957 [454]
Wimpey, J. L.: 1955 [309]
Windmüller, A.: 1942 [47]
Winkel, E.: 1934 [13]; 1935 [17]; 1939 [39]
Wolpe, H.: 1958 [655]
Wright, J. B.: 1953 [155]; 1958 [547]

Yasuura, K.: 1955 [310]

Zaheb, D.: 1954 [253]
Zapletin, N. V.: 1957 [522]
Zemanek, H.: 1955 [311]; 1956 [396]; 1957 [523], [524]; 1958 [656]
Zhozhikashvili, V. A.: 1956 [399], [B-16]
Zhuravlev, Ju. I.: 1958 [657], [658]
Zühlsdorf, W.: 1954 [254], [255]

Subject Index:

List of Periodicals

Am. J. Math.: American Journal of Mathematics, Johns Hopkins University, American Mathematical Society, Johns Hopkins Press, Baltimore 18, Md., USA.
Am. Math. Monthly: American Mathematical Monthly, Mathematical Association of America, University of Buffalo, Buffalo 14, N. Y., USA.
Ann. Comput. Lab. Harvard: The Annals of the Computation Laboratory of Harvard University, Harvard University Press, Cambridge 39, Mass., USA.
Ann. Math.: Annals of Mathematics, Box 231, Princeton, N. J., USA.
Ann. télécommun.: Annales des télécommunications, Centre National d'Études des Télécommunications, Société de la Revue d'Optique, 3 et 5 Boulevard Pasteur, Paris 15, France.
Aplikace Mat.: Aplikace Matematiky, Československá Akademie Věd. Matematiky Ústav, Zitná 25, Prague II, Czechoslovakia.
Arch. elektrischen Übertr.: Archiv der elektrischen Übertragung, S. Hirzel Verlag, Stuttgart, Germany.
Arch. Elektrotech.: Archiv für Elektrotechnik, Springer-Verlag, Reichpietschufer 20, Berlin W. 35, Germany.
Arch. Tech. Messen: Archiv für Technisches Messen und Industrielle Messtechnik, R. Oldenbourg Verlag, Rosenheimerstr. 145, Munich 8, Germany.
ATE J.: Automatic Telephone and Electric Journal, Automatic Telephone and Electric Company Ltd., Strowger House, Arundel St., London W.C.2, England.
Automation and Remote Control: Automation and Remote Control (transl. of Russian, *Avtomat, i Telemekh.*) Instrument Society of America, 313 Sixth Ave., Pittsburgh 22, Pa., USA.
Automatisace: Automatisace, Časopis pro teorii a praxi, Spálená 51, Prague II, Czechoslovakia.
Automatisme: Automatisme, Dunod et Cie., 92 Rue Bonaparte, Paris 6, France.
Avtomat. i Telemekh.: Avtomatika i Telemekhanika, Otdelenije Tekhnicheskikh Nauk, Akademija Nauk SSSR, Kalanchevskaja ul. 15 a, Moscow, USSR. (Transl., see *Automation and Remote Control.*)
Avtomatyka (Kiev): Avtomatyka, Kiev Vydavnytstve Akademii Nauk, Akademija Nauk Ukrainskoj R.S.R., Instytut Elektrotekhniky, Kiev, Ukrainskoj R.S.R.

Bell Labs. Rec.: Bell Laboratories Record, Bell Telephone Laboratories, Inc., 463 West St., New York 14, N. Y., USA.
Bell Sys. Tech. J.: Bell System Technical Journal, American Telephone and Telegraph Co., 195 Broadway, New York 6, N. Y., USA.
Bol. Soc. Math. Mex.: Boletin de Sociedad Mathemática de Mexico, Sociedad Mathemática Mexicana, Tacuba Núm. 5, Mexico.
Brown Boveri Co. Nachr.: Brown Boveri Co. Nachrichten, Mannheim, Germany.
Bull. acad. polon. sci.: Bulletin de l'Académie Polonaise des Sciences, Série des sciences techniques, Prasa i Ksiazka, Warsaw 10, Poland.
Bull. Am. Math. Soc.: Bulletin of the American Mathematical Society, 190 Hope St., Providence 6, R. I., USA.
Bull. assoc. suisse élec.: Bulletin de l'Association Suisse des Électriciens, Seefeldstr. 301, Zurich 8, Switzerland.
Bull. Math. Biophys.: Bulletin of Mathematical Biophysics, University of Chicago Press, 5750 Ellis Ave., Chicago 37, Ill., USA.
Bull. soc. roy. sci. Liège: Bulletin de la Société Royale des Sciences de Liège, L'Université, 7 Place du 20 Août, Liège, Belgium.

Calc. aut. y cib.: Calculo automatico y cibernetica, Sociedad Espanola de Cibernetica, Conde de Penalver 19, Madrid, Spain.
Can. J. Math.: Canadian Journal of Mathematics, Canadian Mathematical Congress, University of Toronto, Toronto, Canada.
Cienc. y Tecnol.: Ciencia y Tecnologia, Departamento de Asuntes Culturales, Union Pan Americana, Washington 6, D. C., USA.
Commun. News: Communication News, N. V. Philips Telecommunicatie Industrie, Hilversum, The Netherlands.
Compt. rend. acad. sci. France: Comptes Rendus Hebdomadaires des Seances de l'Académie des Sciences, Gauthier-Villers, Quai des Grands Augustins 55, Paris 6, France.

Deut. Elektrotech.: Deutsche Elektrotechnik, Zeitschrift für Elektromaschinenbau, Licht- und Messtechnik (now, *Elektrie*), VEB Verlag Technik, Berlin NW. 7, Unter den Linden 12, Germany.
Dokl. AN SSSR: Doklady Akademija Nauk SSSR, Podsosenki per. 21, Moscow B-64, USSR.
Dopovidi. AN URSR: Dopovidi Akademija Nauk Ukrainskoji Radjanskoji Respubliki SSSR., Kiev, Ukrainskoj RSR, USSR.

Elec. Engrg.: Electrical Engineering, American Institute of Electrical Engineers, 33 West 38 St., New York 18, N. Y., USA.
Elec. Mf.: Electrical Manufacturing, Gage Publishing Co., 1250 Sixth Ave., New York 20, N. Y., USA.
Elec. Tech. USSR: Electric Technology U.S.S.R. (transl. of Russian *Elektrichestvo*), Pergamon Press Ltd., 122 East 55 St., New York 22, N. Y., USA.
Electronic Engrg.: Electronic Engineering, Morgan Brothers Ltd., 28 Essex St., Strand, London W.C. 2, England.
Electronics: Electronics, McGraw-Hill Book Co., Inc., 330 West 42 St., New York 36, N. Y., USA.
Electronics World: Electronics World (formerly *Radio News, Radio and Television News*), Ziff-Davis Publishing Co., 434 S. Wabash Ave., Chicago 5, Ill., USA.
Electrotechnica: Electrotechnica, Associatiei Stiintifice a Inginerilor si Technicienilor din R.P.R., Str. Ion Ghica 3, Bucharest, Romania.
Elektrichestvo: Elektrichestvo, Akademija Nauk S.S.S.R., Gosenergoizdat, B. Cherkasski per. 2, Moscow K-12, USSR. (For transl., see *Elec. Tech. USSR.*)
Elektrie: Elektrie, Zeitschrift für die Sozialistische Elektroindustrie, Starkstromstechnik (formerly *Deut. Elektrotech.*), V. E. B. Verlag Technik, Berlin NW 7, Unter den Linden 12, Germany.
Elektronik: Elektronik, Franzio-Verlag, Karlstrasse 35, Munich 37, Germany.
Elektron. Rundschau.: Elektronische Rundschau (formerly *Funk und Ton*), Eichborndamm 141–167, Verlag für Radio-Foto-Kino-Technik G.m.b.H., Berlin-Borsigwalde, Germany.
Elektrosvjaz': Elektrosvjaz', Naucho-Tekhnicheski Zhurnal, Svjaz'-izdat, Ul. Gor'kovo 7, Moscow K-9, USSR. (For transl., see *Telecommun.*)
Elektrotech. obzor: Elektrotechniky Obzor, Krakovská u. č. 8, Prague II, Czekoslovakia.
Elektrotech Z.: Elektrotechnisches Zeitschrift E.T.Z., V.D.E.-Verlag G.m.b.H., Bismarckstr. 33, Berlin-Charlottenburg 2, Germany.
Elektrotekn. Tidsskr. ETT: Elektroteknisk Tidsskrift, Norske Elektrisitetsverkers Forening, Blindern, Oslo, Norway.
Elektrotech. u. Maschinenbau: Elektrotechnik und Maschinenbau, Elektrotechnischer Verein Österreichs, Eschenbachgasse 9, Vienna 1, Austria.
Ericsson Technics: Ericsson Technics, L. M. Ericsson Co., Stockholm 32, Sweden.

Fernmeldetech. Fachber.: Fernmeldetechnische Fachberichte (now *Nachrichtentechnische Fachberichte*), Beihefte der Fernmeldetechnische Zeitschrift (now *Nachrichtentechnische Zeitschrift*), Friedrich Vieweg and Sohn, Burgplatz 1, Braunschweig, Germany.
Fernmeldetech Z.: Fernmeldetechnische Zeitschrift (now *Nachrichtentechnische Zeitschrift N.T.Z.*), Friedrich Vieweg and Sohn, Burgplatz 1, Braunschweig, Germany.
Funk und Ton: Funk und Ton (now *Elektronische Rundschau*), Verlag für Radio-Foto-Kino-Technik G.m.b.H., Eichborndamm 141, Berlin-Borsigwalde, Germany.

Gaz. Matem.: Gazeta de Matemática, Rua Diário de Notícias 134–1°, Lisbon 2, Portugal.

Illustr. Sci. Monthly: Illustrated Science Monthly, London, England.
Illustr. Sci.: Illustrazione Scientifica, Milan, Italy.
Inform. and Contr.: Information and Control, Academic Press, Inc., 111 Fifth Ave., New York 3, N. Y., USA.
Ingegnere: L'Ingegnere, Associazione Nazionale Ingegneri ed Architetti Italiani, Via Dellamercede 33, Rome, Italy.

Ing. ferroviaria: Ingegneria Ferroviaria, Rivista dei Trasporti, Piazza Croce Rossa, Rome, Italy.
Ingenieur: De Ingenieur, Koninklijk Instituut van Ingenieurs, 23 Prinsessegracht, The Hague, Netherlands.
Ingeniören: Ingeniören (Internatl. ed. C.) Engineering House, 27–31 V. Farimagsgade, Copenhagen 5, Denmark.
IRE National Convention Record: IRE National Convention Record, Institute of Radio Engineers, 1 East 79 St., New York 21, N. Y., USA.
IRE Trans.: IRE Transactions (28 different Professional Groups), Institute of Radio Engineers, 1 East 79 St., New York 21, N. Y., USA.
IBM J. Res. and Dev.: I.B.M. Journal of Research and Development, International Business Machine Corp., 590 Madison Ave., New York 22, N. Y., USA.
Instr. and Autom.: Instruments and Automation (now *Instruments and Control Systems*), Instruments Publishing Co., Inc., 845 Ridge Ave., Pittsburgh 12, Pa., USA.
Instr. and Contr. Sys.: Instruments and Control Systems, (formerly, *Instruments and Automation*), Instruments Publishing Co., Inc., 845 Ridge Ave., Pittsburgh 12, Pa., USA.
Instr. Practice: Instrument Practice, Automation and Electronics, United Trade Press Ltd., 9 Gough Square, Fleet St., London E.C. 4, England.
Iz. AN SSSR, Ser. energet. i avtomat.: Izvestia Akademija Nauk SSSR, Otdelenie Tekhnicheskikh Nauk, energetika i avtomatika, Podsosenskij per 21, Moscow B-64, USSR.
Izv. AN SSSR, Ser. fiz.: Izvestia Akademija Nauk SSSR, Otdelenie Tekhnicheskikh Nauk, Serija fizicheskaja, Podsosenskij per 21, Moscow B-64, USSR.

J. Assoc. Comput. Mach.: Journal of the Association for Computing Machinery, 2 East 63 St., New York 21, N. Y., USA.
J. Brit. IRE: Journal of the British Institution of Radio Engineers, 9 Bedford Square, London W.C. 1, England.
J. Comput. Sys.: Journal of Computing Systems, The Institute of Applied Logic, 3101 East 42 St., Minneapolis 6, Minn., USA.
J. Electronics Contr.: Journal of Electronics and Control, Taylor and Francis Ltd., Red Lion Court, Fleet St., London E.C. 4, England.
J. Franklin Inst.: Journal of the Franklin Institute, Franklin Institute of the State of Pennsylvania, Philadelphia 3, Pa., USA.
J. Indian Math. Soc.: Journal of the Indian Mathematical Society, S. Mahadevan 2, Bhaskarapuram, Madras 4, India.
J. Industr. Engrg.: Journal of Industrial Engineering, American Institute of Industrial Engineers, Inc., A. French Bldg., 225 North Ave. N.W., Atlanta, Ga.
J. Inst. Elec. Commun. Engrs. Japan: Journal of the Institute of Electrical and Communication Engineers of Japan, 2–8 Fujimicho, Chiyoda-ku, Tokyo, Japan.
J. IEE: Journal of the Institution of Electrical Engineers, Savoy Place, London W.C. 2, England.
J. Inst. Telecommun. Engrs.: Journal of the Institution of Telecommunication Engineers, Box 481, New Delhi, India.
J. Math. Phys.: Journal of Mathematics and Physics, Technology Press, Massachusetts Institute of Technology, Cambridge 39, Mass., USA.
J. Soc. Industr. Appl. Math.: Journal of the Society for Industrial and Applied Mathematics, P. O. Box 7541, Philadelphia 1, Pa., USA.
J. Symbolic Logic: Journal of Symbolic Logic, The Association for Symbolic Logic, Inc., 190 Hope St., Providence 6, R.I., USA.

Machine Design: Machine Design, Penton Publishing Co., Penton Bldg., Cleveland 13, Ohio, USA.
Math. Comput.: Mathematics of Computation (formerly *Mathematical Tables and Other Aides to Computation*), National Academy of Sciences, Printing and Publishing Office, 2101 Constitution Ave., Washington 25, D.C., USA.
Math. Rev.: Mathematical Reviews, American Mathematical Society, 190 Hope St., Providence 6, R. I., USA.
Math. Table Other Aids Comp.: Mathematical Tables and Other Aides to Computation (now *Math. Comput.*), National Academy of Sciences, Printing and Publishing Office, 2101 Constitution Ave., Washington 25, D. C., USA.
Mem. Fac. Engrg. Kyushu Univ.: Memoirs of the Faculty of Engineering, Kyushu Imperial University, Fukuoka, Japan.
Mem. Fac. Engrg. Nagoya Univ.: Memoirs of the Faculty of Engineering, Nagoya University, Fuo-cho, Chikusa-ku, Nagoya, Japan.
Mém. soc. ing. civils France: Mémoirs de la Société des Ingenieurs Civils de France, 19 rue Blanche, Paris 9, France.
Monatsh. Math.: Monatshefte für Mathematik, Springer-Verlag, Molkerbastei 5, Vienna 1, Austria.

Nachrtech. Fachber.: Nachrichtentechnische Fachberichte N.T.F. (formerly *Fernmeldetechnische Nachberichte*), Beihefte der Nachrichtentechnische Zeitschrift, Friedrich Vieweg and Sohn, Burgplatz 1, Braunschweig, Germany.
Nachrtech Z.: Nachrichtentechnisches Zeitschrift (formerly *Fernmeldetechnische Zeitschrift*), Friedrich Vieweg and Sohn, Burgplatz 1, Braunschweig, Germany.
Nebraska Blue Print: Nebraska Blue Print, Engineering Society of the University of Nebraska, Lincoln, Neb., USA.
Nippon Elec. Commun. Engrg.: Nippon Electrical and Communication Engineering, Institute of Telegraph and Telephone Engineers of Japan, No. 1447 Tokyo-Kaijo, Marunouchi, Tokyo, Japan.
Nuclear Engrg.: Nuclear Engineering, Temple Press, Bowling Green Lane, London E.C. 1, England.

Onde Elec.: L'Onde Électrique, Société des Radioélectriciens, 10 Ave. Pierre-Larousse, Malakoff (Seine), Paris, France.
Org. Fortschr. Eisenbahnwes.: Organ für die Fortschritte des Eisenbahnwesens in technischer Beziehung, Springer-Verlag, Berlin, Germany.
Österr. Z. Elek. Wirtsch.: Österreichische Zeitschrift für Elektrisitätswirtschaft, Springer-Verlag, Mölkerbastei 5, Vienna 1, Austria.

Philips Res. Repts.: Philips Research Reports, N.V. Philips Gloeilampenfabrieken, Eindhoven, Netherlands.
Phil. Sci.: Philosophy of Science, The Philosophy of Science Association, Williams and Wilkins Co., 428 East Preston St., Baltimore, 2, Md., USA.
Portugalaie Mathem.: Portugaliae Mathematica, Sociedade Portuguesa de Matematica, Rua Nova de Trinidade 1, Lisbon, Portugal.
Poste e Telecommun.: Poste e Telecommunicazioni (formerly *Rassegna Technica Mensile Post e Telecommunicazione*), Via della Vite 107, Rome, Italy.
PTT-Bedrijf: P.T.T.-Bedrijf, Direction Centrale des P.T.T., 12 Kortenaerkade, The Hague, The Netherlands.
Priborostroenie: Priborostroenie, Sovet Ministrov SSSR, Gosudarstvennyj Naucho-tekhnicheski Komitet, Ul'Gorkova 6, Moscow K-9, USSR.
Prikl. Mat. Mekh.: Prikladnaja Matematika i Mekhanika, Akademija Nauk SSSR, Otdelenie Tekhnicheskikh Nauk, Institut Mekhaniki, Kalanchevskaja ul. 15a, Moscow, USSR. (For English transl., see *Applied Mathematics and Mechanics.*)
Proc. Cambridge Phil. Soc.: Proceedings of the Cambridge Philosophical Society (Mathematical and Physical Sciences),Cambridge University Press, 200 Euston Rd., London N.W. 1, England.
Proc. IRE: Proceedings of the Institute of Radio Engineers, 1 East 79 St., New York 21, N. Y., USA.
Proc. Natl. Acad. Sci.: Proceedings of the National Academy of Science of the United States, Journals Division, University of Chicago Press, 5750 Ellis Ave., Chicago 37, Ill., USA.
Przeglad Telekom.: Przeglad Telekomunikacyiny, Sekcji Telekomunikacyjenj Stowarzyszenia Elektryków Polskich, Wydawnictwa Czasopism Technicznych N.O.T., Barbary 2, Warsaw, Poland.

Quart. Appl. Math.: Quarterly of Applied Mathematics, Brown University, Providence 12, R. I., USA.

Radio-Electronics: Radio-Electronics, Gernsback Publications Inc., 154 West 14 St., New York 11, N. Y., USA.
Radio-Mentor: Radio-Mentor, Hubertusbader Str. 16, Berlin-Grünewald, Germany.
Radio and Tele. News: Radio and Television News, (formerly, *Radio News*, now, *Electronics World*), Ziff-Davis Publishing Co., 434 S. Wabash Ave., Chicago 5, Ill., USA.
Rend. mat. e appl.: Rendiconti di matematica e delle sue applicazione, Università di Roma, Instituto Nazionale di Alta Matematica, Rome, Italy.

Rev. cienc. apl.: Revista de ciencia aplicada, Apartado de Correos 743, Madrid, Spain.
Rev. Telecommun: Revista de Telecomunicación, Direccion General de Corres y Telecomunicación, Palacio de Comunicaciónes, Madrid, Spain.
Rev. telegr. electrón.: Revista telegrafica, electronica, Avda. Martin Garcia 653, Buenos Aires, Argentina.
Rev gén. élec.: Revue générale de l'électricité, 12 Place Henri-Bergson, Paris 8, France.
Rev. H. F.: Revue H.F., Société belge des ingenieurs des télécommunications et d'Électronique, 55 Rue Defacqz, Brussels, Belgium.

Science: Science, American Association for the Advancement of Science, 1515 Mass. Ave., N.W. Washington 5, D. C., USA.
Sci. Elec.: Scientia Electrica, A. G. Fachschriften-Verlag and Buchdruckerei, Zurich, Switzerland.
Sci. Sinica: Scientia Sinica, Academia Sinica, Guozi Shudion, 38 Suchou Hutung, Peking, China.
Signal u. Draht: Signal und Draht (formerly *Zeitschrift für das gesamte Eisenbahnwesen Sicherungs und Fernmeldewesen*), Berlin Germany.
Slaboproudý obzor: Slaboproudý Obzor, Krakovska 8, Prague 2, Czekoslovakia.
S. African Elec. Rev.: South African Electrical Review, 66 Commissioner St., Johannesburg, South Africa.
Soviet Phys.—Tech. Phys.: Soviet Physics—Technical Physics (transl. of Russian, *Zhurnal Tekhnicheskoj Fiziki*), American Institute of Physics, 335 East 45 St., New York 17, N. Y., USA.
Sperry Engrg. Rev.: Sperry Engineering Review, Engineering Division of Sperry Gyroscope Co., Division of Sperry Corporation, Great Neck, N. Y., USA.
Strument. e Autom.: Strumentazione e Automazione, Via Marcona 15, Milan, Italy.
Studi si Cercetari: Studii si Cercetări de Energetică, Academia Republicii Populare Romine, Institutul de Energetica, Sos. Vitan nr. 236, Bucuresti, Romania.

Técnica: Técnica, Rua Cruz dos Poiais 103, Lisbon, Portugal.
Tekh. Zheleznykh Dorog: Tekhnika Zheleznykh Dorog, (Railroad Engineering), Moscow, USSR.
Tek. Fören. i Finland Förh.: Tekniska Föreningens i Finland Förhandlingar (now *Teknisk Forum*), Svenska Teaterhuset, Helsinki, Finland.
Tek. Tidskr.: Teknisk Tidsskrift, Box 841, Stockholm 1, Sweden.
Tek. Ukeblad: Teknisk Ukeblad, Den Norske Ingeniørforening, Kronprinsensgt. 9, Oslo, Norway.
Telecommun.: Telecommunications (English transl. of Russian, *Elektrosvjaz'*), Pergamon Press, 122 East 55 St., New York 22, N. Y., USA.

Trans. AIEE: Transactions of the American Institute of Electrical Engineers (3 pts.), 33 West 39 St., New York 18, N. Y., USA.
Trans. Am. Math. Soc.: Transactions of the American Mathematical Society, 190 Hope St., Providence 6, R. I., USA.
Trans. Ill. State Acad. Sci.: Transactions of the Illinois State Academy of Science, Springfield, Ill., USA.
Trans. S. African Inst. Elec. Engrs.: Transactions of the South African Institute of Electrical Engineers, Kelvin House, Marshal and Holland St., Johannesburg, South Africa.
Trudy Leningrad Eksper. Elektrotekh. Labor: Trudy Leningradskogo Eksper. Elektrotekhnicheskogo Labor., Leningrad, USSR.
Trudy Minsk. Vyssh.: Trudy Minskogo Vysshegoradiotekhnicheskogo Inzhenernogo Uchilishcha (Transactions of Minsk Higher Radiotech. Engrg. School), Minsk, USSR.

Uchenye Zapiski Kishch. G. U.: Uchenye Zapiski Kishchinevskogo Gosudarstvennogo Universiteta (Scientific notes of the Kishinev State University) Kishchinev, USSR.
Uchenye Zapiski Moskov G. U.: Uchenye Zapiski Moskovskogo Gosudarstvennogo Universiteta (Scientific notes of the Moscow State University), Moscow, USSR.
Univ. Calif. Publ. Math.: University of California Publications in Mathematics, University of California Press, Berkeley, Calif., USA.
Uspekhi Mat. Nauk SSSR: Uspekhi Matematicheskikh Nauk SSSR, Akademija Nauk SSSR, Fizmatgiz, Leninskij Prospekt 15, Moscow V-71, USSR.

Vychisl. Mat. Tekh.: Vychislitel'naja Matematika i Vychislitel'naja Tekhnika, Akademija Nauk SSSR., Institut Tochnoj Mekhaniki i Vychislitel'noj Tekhniki, Moscow, USSR.

Western Elec. Engr.: Western Electric Engineer, Western Electric Co., Inc., 195 Broadway, New York 7, N. Y., USA.

Z. angew. Math. Mech.: Zeitschrift für angewandte Mathematik und Mechanik, Ingenieurwissenschaftliche Forschungsarbeiten, Akademie-Verlag G.m.b.H., Mohrenstrasse 39, Berlin W. 8, Germany.
Z. angew. Math. u. Phys.: Zeitschrift für angewandte Mathematik und Physik, Verlag Birkhäuser, Elisabethenstrasse 15, Basel 10, Switzerland.
Z. math. Logik Gr. Math.: Zeitschrift für mathematischen Logik und Grundlagen der Mathematik, Humboldt Universität, Berlin, Deutscher Verlag der Wissenschaften, Niederwallstr. 39, Berlin W. 8, Germany.
Z. Österr. Ing. Arch. Verrein: Österreichischer Ingenieur- und Architekten-Vereines, Zeitschrift, Springer-Verlag, Mölkerbastei 5, Vienna 1, Austria.
Zhur. Tekh. Fiz.: Zhurnal Tekhnicheskoj Fiziki SSSR, Akademija Nauk SSSR, Mendelejevskaja lin. 1, Leningrad V-164, USSR.

List of Frequently Used Russian Abbreviations[2]

AN SSSR	Akad. Nauk SSSR, Academy of Sciences, U.S.S.R.
Dokl.	Doklady, Report, paper
GEI	Gos. Energ. Izd., State Power Press
GFKI	Gos. Fiz. Khim. Izd., State Physical Chemistry Press, State Scientific and Technical Press
GITTL	Gos. Izd. Tekh. Teor. Lit., State Press for Technical and Theoretical Literature
GTI	Gos. Tekh. Izd., State Technical Press
GTTI	Gos. Tekh. Teor. Izd., State Technical and Theoretical Press
IAT	Inst. Avtomat. Telemek., Institute of Automation and Remote Control
ILI	Inostr. Lit. Izd., Foreign Literature Press
IPM	Inst. Primen. Matem., Institute of Applied Mathematics
ISN	Izd. Sov. Nauk, Soviet Science Press
Izd.	Izdatel'stvo, Press (Publishing House)
Izv.	Izvestija, Proceedings, transactions, new
LETI	Leningrad Elek. Tekh. Inst., Leningrad Electrotechnical Institute
LFTI	Leningrad Fiz. Tekh. Inst., Leningrad Institute of Physics and Technology
MGU	Moskva Gos. Univ., Moscow State University
Sb.	Sbornik, Collection (of articles)
SI	Svjaz'izdat., United Press
Trudy	Work (scientific work)
TSI	Tipogr. Svjaz. Izdat., Typogr. Printing House
Usp.	Uspekhi, Progress (report)

[2] A more complete list of abbreviations, from which a major part of this is taken, may be found on the inside back cover of *Automation and Remote Control.*

An Algorithm for Rapid Binary Division*

J. B. WILSON†, ASSOCIATE, IRE, AND R. S. LEDLEY‡, MEMBER, IRE

Summary—A new algorithm is given for reducing the number of additions and subtractions required in binary division in a computer. The algorithm is presented in two parts. A simplified algorithm, which can significantly reduce the number of operations with minimal additional circuitry, is used to develop the justification of the method. The complete algorithm introduces modifications which allow the minimum number of operations by examination of no more than the leading five bits of the divisor and remainder. An average of two-thirds can be saved in the number of operations.

Introduction

A BINARY number can be thought of as composed of strings of units, strings of zeros, strings of units including isolated zeros, and strings of zeros including isolated units. For example, the binary number in Fig. 1 has a string of units from the 2^{-1} to 2^{-4} positions, a string of zeros from the 2^{-5} to 2^{-8} positions, a string of units from the 2^{-9} to 2^{-13} positions with an isolated zero at the 2^{-11} position, and a string of zeros from the 2^{-14} to 2^{-18} positions with an isolated unit at the 2^{-16} position. The significance of this string-type decomposition of a binary number depends upon three elementary observations: 1) a string of units from 2^{-p} to 2^{-q} positions contributes a value of $(2^{-p+1}-2^{-q})$ to the magnitude of the number, for example $0.11111 = 1-0.00001 = 2^0-2^{-5}$; 2) an isolated unit in the 2^{-p} position in a string of zeros contributes 2^{-p}, for example, $0.00100 = 2^{-3}$; 3) an isolated zero in the 2^{-q} position decreases the magnitude of the number of 2^{-q}, for example, $0.11011 = 0.11111 - 0.00100 = 2^0 - 2^{-3} - 2^{-5}$. Altogether, then, we can write the binary number of Fig. 1 as

$$0.111100001101100100 = 2^0 - 2^{-4} + 2^{-8} - 2^{-11} - 2^{-13} + 2^{-16}.$$

A convenient shorthand notation is to place "+" or "−" above those positions of a binary number whose values respectively contribute to or decrease the magnitude of the number. Thus, would write for the number of Fig. 1

$$\overset{+}{0}.111\bar{0}000\overset{+}{0}11\bar{0}1\bar{1}00\overset{+}{1}00.$$

The generality of these elementary observations is clear.[1]

Smith and Weinberger[2] utilized such a decomposition of binary numbers in developing a technique of rapid multiplication, for only those additions to or subtractions from the partial product need be performed that are indicated by the decomposition of the multiplier. For example,

$$(0.1101)(\overset{+}{0}.11\bar{0}1\bar{1}00\overset{+}{1}00) = (0.1101)(2^0 - 2^{-3} - 2^{-5} + 2^{-8})$$

requires only four operations. Clearly there can be different decompositions for the same nonzero binary number. Rules for obtaining a "best" decomposition, in the sense of having the least number of terms, are given by Smith and Weinberger (discussed by Ledley[1]), Lehman,[3] Tocher,[4] and Reitwiesner.[5] Following Smith and

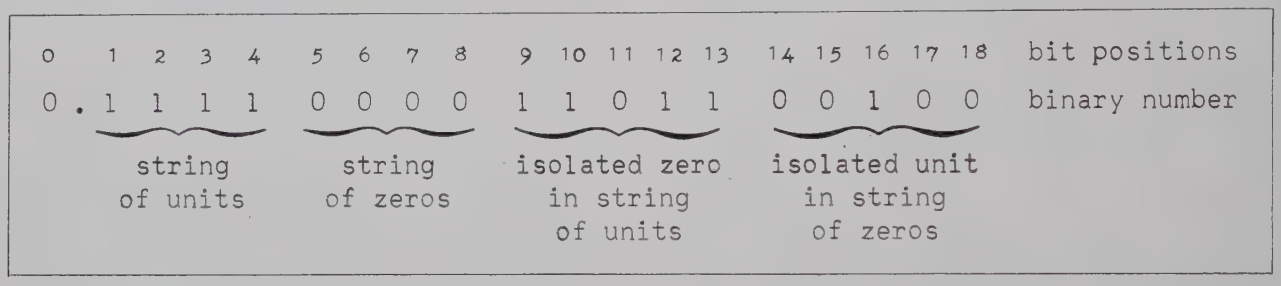


Fig. 1—Illustration of the decomposition of a binary number in terms of strings of units and zeros, with isolated zeros and units.

* Received by the PGEC, November 25, 1960; revised manuscript received, May 26, 1961. The research in this paper was supported by the Information Systems Branch, U. S. Office of Naval Research, under Contract No. Nonr. 3265(00) with the National Biomedical Research Foundation.

† National Biomedical Research Foundation, Silver Spring, Md. Formerly with the School of Engineering, The George Washington University, Washington, D. C.

‡ National Bio-medical Research Foundation. Formerly with the National Bureau of Standards, Washington, D. C.

[1] For a more detailed discussion of this decomposition and its applications to rapid arithmetic, see R. S. Ledley, "Digital Computer and Control Engineering," McGraw-Hill Book Co., Inc., New York, N. Y., ch. 16; 1960.

[2] J. L. Smith and A. Weinberger, "Shortcut Multiplication for Binary Digital Computers," *NBS Circular 591*, Sec. 1, pp. 13–22.

[3] M. Lehman, "High speed multiplication," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-6, pp. 204–205; September, 1957.

[4] K. D. Tocher, "Techniques of multiplication and division for automatic binary computers," *Quart. J. Mech. and Appl. Math.*, vol. 11, pp. 364–384; August, 1958.

[5] G. W. Reitwiesner, "Binary arithmetic," in "Advances in Computers," F. L. Alt, Ed., Academic Press Inc., New York, N. Y.; 1960.

Weinberger, two or more adjacent zeros should be considered as a string; two or more adjacent units as a string; cases such as 0101010 should be considered as $\overset{+\ +\ +}{0101010}$ rather than $\overset{+\ -\ -\ -}{0101010}$, that is, as isolated units rather than as isolated zeros in a string of units; and so forth. For rapid multiplication, a best decomposition must be obtained. The method of rapid division to be described essentially reverses this rapid multiplication process.

The Two Algorithms

We shall present first a simplification of the algorithm, in order to explain and justify the process more easily. This partial algorithm has the advantage of extreme simplicity and in a majority of cases significantly reduces the number of operations required for division. Finally, we shall give the modifications necessary to take full advantage of the method, presenting the complete algorithm that produces the minimum number of operations for division.

The relation between our algorithm and others previously given in the literature is as follows: Our partial algorithm gives an explicit form for binary numbers of a procedure that is suggested by Tocher.[4] Our complete algorithm achieves maximum efficiency in reducing the number of additions and subtractions to a minimum. Robertson[6] allows alternative decomposition of the quotient to accommodate at most one addition or subtraction for each two binary positions and exhibits sets of comparisons of the leading digits of the divisor and current remainder by which his process is controlled. We allow alternative decomposition, within narrower limits, to yield a "best" decomposition, and we exhibit the precise configurations of leading bits of the divisor and remainder which must be examined to conform to our limits. Reitwiesner[5] develops a "canonical" minimum decomposition for the quotient, and demonstrates how to obtain it by comparisons made at each bit of the quotient; but his comparisons are made over the full lengths of the divisor and remainder. Tocher also requires comparison of the full lengths of the divisor and remainder, by virtue of his employment of the factor 2/3. Other writers using different approaches have, insofar as we know, been unable to achieve maximum efficiency. The principal factor in the development of our algorithm is our acceptance of alternative "best" decompositions whenever convenient to do so; thus, we do not necessarily employ the "canonical" form developed in most of the cited references.

The Simplified Algorithm

The procedure is summarized by Fig. 2;[7] the procedure ends when i reaches the number of bits desired in the quotient. We assume that the binary point is to the far left of each word, that the denominator D is positive and normalized (its most significant bit a unit), that the numerator N is positive, and that $N<D$, with N either normalized or with a single zero after the binary point. The first step is to form the first remainder, $N'=N-D$. Since, with our initial assumptions, N' is negative, the negative loop of Fig. 2 is followed. If N' has α zeroes to the right of the binary point, then the quotient Q has at least $\alpha-1$ units to the right of the point. (In this initial step, $Q_i=0$ for $i=0$ means merely that $Q<1$.) Now N' is normalized and the second remainder, $N''=N'+D$, is formed. If N'' is negative, then $Q_{0+\alpha}=0$, and the following bits are units. If N'' is positive, then $Q_{0+\alpha}=1$, and the following bits are zeros. The procedure continues in this way, differencing and normalizing each time, and determining Q_i and Q_{i+1} through $Q_{(i+\alpha)+1}$ at each step. For example, consider $N=.1001\ 1111\ 0000\ 1100$ and $D=.1101$:

		quotient Q:
N:	.1 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0	
$-D$:	− .1 1 0 1	
N':	−×0 0.1 1 0 0 0 0 1 1 1 1 0 1 0 0	0.1?
$+D$:	+ .1 1 0 1	
N'':	+×0 0 0 0.1 1 0 0 0 0 1 1 0 0	.1100 0?
$-D$:	− .1 1 0 1	
N''':	−×0 0 0 0.1 1 0 1 0 0	.1100 0011 1?
$+D$:	+ .1 1 0 1	
N^{iv}:	+×0 0 0 0 0 0 0 0	.1100 0011 1100 · · · .

[6] J. E. Robertson, "A new class of digital division methods," IRE Trans. on Electronic Computers, vol. EC-7, pp. 218–222; September, 1958.

[7] The complete algorithm also follows Fig. 2, with the step normalizing $N^{(s)}$ modified by Rules I and II given below.

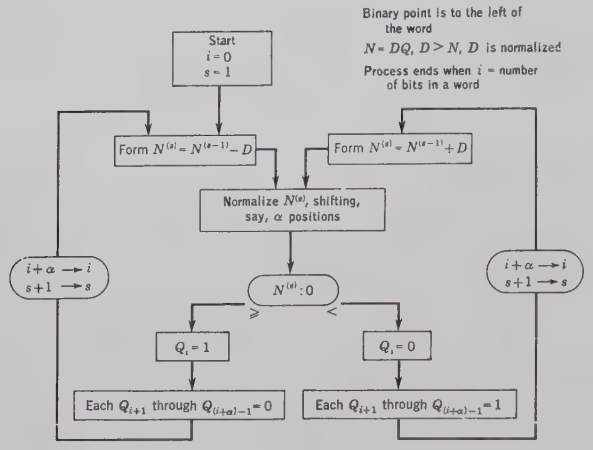


Fig. 2—Flow chart for rapid binary division (see footnote 7).

As a further example consider $N=.1011\ 0000\ 0110$ and $D=.1101$:

$$\begin{array}{rl} N: & +\ .1\ 0\ 1\ 1\quad 0\ 0\ 0\ 0\quad 0\ 1\ 1\ 0 \\ -D: & -\ .1\ 1\ 0\ 1 \\ \hline N': & -\ {}_{\times}0\ 0\ 0.1\quad 1\ 1\ 1\ 1\quad 1\ 0\ 1\ 0 \\ +D: & \quad +\ .1\quad 1\ 0\ 1 \\ \hline N'': & \quad -\ {}_{\times}0\quad 0.1\ 0\ 1\quad 1\ 0\ 1\ 0 \\ +D: & \qquad +\ .1\ 1\ 0\quad 1 \\ \hline N''': & \qquad +\ {}_{\times}0\ 0\ 0\quad .1\ 1\ 1\ 0 \\ -D: & \qquad\qquad -.1\ 1\ 0\ 1 \\ \hline N^{\mathrm{iv}}: & \qquad\qquad +{}_{\times}0\ 0\ 0.1 \end{array}$$

quotient Q:

$$\begin{array}{lll} 0.1\ 1? & & \\ \downarrow & & \\ ..1101 & ? & \\ & \downarrow & \\ .1101 & 100? & \\ & \downarrow & \\ .1101 & 1001 & 00?\ \cdots. \end{array}$$

In this example the first zero and the fifth unit were isolated.

Rationale for the Algorithm

Some preliminary elementary observations will aid the subsequent discussion. First note that

$$1-2^{-\gamma}=.\overset{1}{1}\overset{2}{1}\cdots 1\overset{\gamma}{1} \tag{1}$$

but that if $0<R<1$, then

$$1-2^{-\gamma}(1+R)=.\overset{1}{1}\overset{2}{1}\cdots 1\overset{\gamma}{0}\cdots. \tag{2}$$

Next, observe that if $2^{-1}\leq D<1$ and

$$M=DX,\quad \text{where}\quad M=.\overset{1}{0}\overset{2}{0}\cdots 0\overset{\gamma}{1}\cdots \tag{3}$$

then

$$X=.\overset{1}{0}\overset{2}{0}\cdots 0\left\{\begin{matrix}\overset{\gamma}{0}\\1\end{matrix}\right\}+2^{-\gamma}X' \tag{4}$$

with $2^{-\gamma}X'$, *i.e.*, the rest of X, such that $0<X'<1$—since with $2^{-\gamma-1}\leq M<2^{-\gamma}$ and $2^{-1}\leq D<1$ in $X=M/D$, $(2^{-\gamma-1}/1)<X<(2^{-\gamma}/2^{-1})$, whence $2^{-\gamma-1}<X<2^{-\gamma+1}$.

Similarly if $2^{-1}\leq D<1$ and

$$M=D(X-1),\quad \text{where}\quad M=-.00\cdots 0\overset{\gamma}{0}1\cdots \tag{5}$$

then

$$X=.11\cdots 1\left\{\begin{matrix}\overset{\gamma}{0}\\1\end{matrix}\right\}+2^{-\gamma}X' \tag{6}$$

where $0<X'<1$. We shall frequently refer to these equations.

The initial conditions are $N=DQ$, $D>N>0$, $2N>D$, and D is normalized (*i.e.*, $2^{-1}\leq D<1$). The first step is to form

$$N'=N-D=D(Q-1)$$

where $N'<0$. Suppose that $N'=-.00\cdots\overset{\alpha}{0}01\cdots$, then the conditions of (5) are satisfied and the first $\alpha-1$ bits of Q are found by applying (6):

$$Q=.11\cdots 1\left\{\begin{matrix}\overset{\alpha}{0}\\1\end{matrix}\right\}+2^{-\alpha}Q'$$

where the α bit is as yet undetermined. To proceed with the iteration, we use the index s. The first step has left us with the *general case A:*

$$N^{(s)}<0\quad\text{and}\quad N^{(s)}=D(Q^{(s-1)}-1) \tag{7}$$

where

$$Q^{(s-1)}=.11\cdots 1\left\{\begin{matrix}\overset{\alpha}{0}\\1\end{matrix}\right\}+2^{-\alpha}Q^{(s)},\quad 0<Q^{(s)}<1. \tag{8}$$

We next form

$$N^{(s+1)}=2^{\alpha}N^{(s)}+D=2^{\alpha}D(Q^{(s-1)}-1+2^{-\alpha}). \tag{9}$$

Three subcases may arise, $N^{(s+1)}=0$, $N^{(s+1)}<0$, or $N^{(s+1)}>0$, which are treated as follows:

A1) $N^{(s+1)}=0$: Then by (9), we find $Q^{(s-1)}-1+2^{-\alpha}=0$, or $Q^{(s-1)}=1-2^{-\alpha}$, which determines the α bit as 1; $Q^{(s)}=0$, and the division process is completed.

A2) $N^{(s+1)}<0$: Then by (9), since $D>0$, we find

$$Q^{(s-1)}-1+2^{-\alpha}<0$$

or $Q^{(s-1)}<1-2^{-\alpha}=.11\cdots\overset{\alpha}{1}1$, whence

$$Q^{(s-1)}=.11\cdots\overset{\alpha}{1}0+2^{-\alpha}Q^{(s)},\qquad 0<Q^{(s)}<1$$

which determines the α bit for this subcase. Thus,

$$\begin{aligned}Q^{(s-1)}-1+2^{-\alpha}&=.11\cdots\overset{\alpha}{1}0+2^{-\alpha}Q^{(s)}-1+2^{-\alpha}\\&=1-2^{-(\alpha-1)}+2^{-\alpha}Q^{(s)}-1+2^{-\alpha}\\&=2^{-\alpha}(1-2+Q^{(s)})\\&=2^{-\alpha}(Q^{(s)}-1)\end{aligned}$$

which by (9) gives us

$$N^{(s+1)}=D(Q^{(s)}-1). \tag{10}$$

Suppose that $N^{(s+1)}=-.00\cdots\overset{\beta}{0}1\cdots$, then by (5) the next $\beta-1$ bits are determined, as in (6):

$$Q^{(s)}=.11\cdots1\left\{\begin{matrix}\overset{\beta}{0}\\1\end{matrix}\right\}+2^{-\beta}Q^{(s+1)},\quad 0<Q^{(s+1)}<1. \tag{11}$$

Consider the next successive iteration after subcase A2): Note from (10) and (11) that subcase A2) has left us in the same situation as in the general case A, (7) and (8), with $s+1$ replacing s. Hence the next successive iteration involves forming (9) etc., closing the loop for this subcase.

A3) $N^{(s+1)}>0$: Then by (9), since $D>0$, we find

$$Q^{(s-1)}+2^{-\alpha}>1,$$

or using (1), $Q^{(s-1)}>1-2^{-\alpha}=.11\cdots\overset{\alpha}{1}1$; that is,

$$Q^{(s-1)}=.11\cdots\overset{\alpha}{1}1+2^{-\alpha}Q^{(s)},\qquad 0<Q^{(s)}<1.$$

Hence the α bit is 1. On using (1) again, we find

$$\begin{aligned}Q^{(s-1)}-1+2^{-\alpha}&=.11\cdots\overset{\alpha}{1}1+2^{-\alpha}Q^{(s)}-1+2^{-\alpha}\\&=2^{-\alpha}Q^{(s)}\end{aligned}$$

which from (9) gives us

$$N^{(s+1)}=DQ^{(s)}. \tag{12}$$

Suppose that $N^{(s+1)}=.00\cdots\overset{\beta}{0}01\cdots$; then the conditions of (3) are satisfied, and by applying (4) the next $\beta-1$ bits are determined:

$$Q^{(s)}=.00\cdots0\left\{\begin{matrix}\overset{\beta}{0}\\1\end{matrix}\right\}+2^{-\beta}Q^{(s+1)},\quad 0<Q^{(s+1)}<1 \tag{13}$$

where the β bit is as yet undetermined.

Consider the next successive iteration after subcase A3). If in (12) and (13) of subcase A3) we replace $s+1$ by s, we have the *general case B* [compare with (7) and (8)]:

$$N^{(s)}>0 \quad\text{and}\quad N^{(s)}=DQ^{(s-1)} \tag{14}$$

$$Q^{(s-1)}=.00\cdots0\left\{\begin{matrix}\overset{\beta}{0}\\1\end{matrix}\right\}+2^{-\beta}Q^{(s)},\quad 0<Q^{(s)}<1. \tag{15}$$

We next form

$$N^{(s+1)}=2^{\beta}N^{(s)}-D=2^{\beta}D(Q^{(s-1)}-2^{-\beta}). \tag{16}$$

Three subcases may arise, $N^{(s+1)}=0$, $N^{(s+1)}<0$, $N^{(s+1)}>0$, which are treated as follows:

B1) $N^{(s+1)}=0$: Then by (16), $Q^{(s-1)}=2^{-\beta}$, which determines the β bit as 1: $Q^{(s)}=0$, and the division process is completed.

B2) $N^{(s+1)}<0$: Then by (16), $Q^{(s-1)}<2^{-\beta}$, or

$$Q^{(s+1)}=.00\cdots\overset{\beta}{0}0+2^{-\beta}Q^{(s)},\qquad 0<Q^{(s)}<1$$

determining the β bit for this case. Thus,

$$\begin{aligned}Q^{(s-1)}-2^{-\beta}&=.00\cdots\overset{\beta}{0}0+2^{-\beta}Q^{(s)}-2^{-\beta}\\&=2^{-\beta}(Q^{(s)}-1)\end{aligned}$$

which from (16) gives us

$$N^{(s+1)}=D(Q^{(s)}-1). \tag{17}$$

Suppose that $N^{(s+1)}=.00\cdots\overset{\gamma}{.}01$, then by (5) and (6), we find the next $\gamma-1$ bits as above. The next successive iteration after subcase B2) clearly follows the general case A, closing the loop in this subcase.

B3) $N^{(s+1)}>0$: Then by (16), $Q^{(s-1)}>2^{-\beta}$, or

$$Q^{(s-1)}=.00\cdots\overset{\beta}{0}1+2^{-\beta}Q^{(s)},\qquad 0<Q^{(s)}<1$$

determining the β bit for this subcase. Thus,

$$Q^{(s-1)}-2^{-\beta}=.00\cdots\overset{\beta}{0}1+2^{-\beta}Q^{(s)}-2^{-\beta}Q^{(s)}$$

from which, by (16), we find

$$N^{(s+1)}=DQ^{(s)}. \tag{18}$$

Suppose that $N^{(s+1)}=.00\cdots\overset{\gamma}{.}01$, then by (3) and (4) we find the next $\gamma-1$ bits as above. The next successive iteration after subcase B3) clearly follows the general case B, closing the loop in this subcase. Since all possibilities are completed, this concludes the discussion.

Bad Cases

Although this simplified algorithm will reduce the number of subtractions required in the majority of cases, for a certain minority of cases normalizing $N^{(s)}$ would result in shifting $N^{(s)}$ either one place too far or not far enough. Consider for example (Case I) two ways of dividing $N=.011001010001$ by $D=.100001$, the first following the flow chart of Fig. 2, the second following the flow chart by choosing Q_i, but shifting $N^{(s)}$ one place less on the second differencing:

```
+ .0 1 1 0 0 1 0 1 0 0 0 1
- .1 0 0 0 0 1
- ×0 0 0.1 1 1 1 0 1 1 1 1     Q = 0.11?
    + .1 0 0 0 0 1                    ↓
    - ×0.1 1 1 0 0 1 1 1          .110?
      + .1 0 0 0 0 1                  ↓
      - ×0.1 1 0 0 0 1 1          .1100?
        + .1 0 0 0 0 1                ↓
        - ×0.1 0 0 0 0 1          .11000?
          + .1 0 0 0 0 1              ↓
            ×0 0 0 0 0 0          .110001
```

```
+ .0 1 1 0 0 1 0 1 0 0 0 1
- .1 0 0 0 0 1
- ×0 0.0 1 1 1 1 0 1 1 1 1     Q = .1?
    + .1 0 0 0 0 1                    ↓
    + ×0 0 0 0.1 0 0 0 0 1        .11000?
        - .1 0 0 0 0 1                ↓
          ×0 0 0 0 0 0            .110001
```

The first required five differencings, the second only three.

Consider also (Case II) two ways of dividing $N=.100010001$ by $D=.1101$, the first following the flow chart of Fig. 2, the second following the flow chart for choosing the Q_i, but shifting $N^{(s)}$ one place further:

```
+ .1 0 0 0 1 0 0 0 1
- .1 1 0 1
- ×0.1 0 0 0 1 1 1 1     Q = 0.?
  + .1 1 0 1                   ↓
  - ×0.1 0 0 0 0 0 0 1      .1?
    + .1 1 0 1                 ↓
    + ×0.1 0 0 1 1 1       .10?
      - .1 1 0 1               ↓
      - ×0 0.1 1 0 1       .1010?
        + .1 1 0 1             ↓
          ×0 0 0 0         .10101
```

```
+ ×1.0 0 0 1 0 0 0 1     Q = .?
  - .1 1 0 1                   ↓
  + ×0 1.0 0 0 0 0 0 1      .10?
    - .1 1 0 1                 ↓
    + ×0 0.1 1 0 1         .1010?
      - .1 1 0 1               ↓
        ×0 0 0 0           .10101
```

The first required five differencings (one for each place of Q), the second only three (one for each unit of Q). In fact if Q had been, say, a 44-bit word alternating ones and zeros, then the first method would have required 43 subtractions, the second only 22.

For the maximum speed of division (*i.e.*, for the minimum number of required subtractions), the algorithm must recognize these and similar cases, and adjust the normalizing accordingly. The above two bad cases illustrate the two categories calling for adjustment: Case I illustrates the failure to recognize the end of a string of units (or zeros) so that $N^{(s)}$ is shifted beyond the end of the string. Case II illustrates the failure to recognize an isolated unit (or zero).

The Complete Algorithm

The most efficient possible method for generating a quotient would be precisely to reverse the rapid multiplication process mentioned above. Consider, for example, the following products $N=DQ$:

```
                  0.1 0 0 0 0 1    D
                  0.1 1̄ 0 0 0 1    Q
                + 0 1 0 0 0 0 1
              - 0 1 0 0 0 0 1
            - 0 0 0 1 1 1 1 0 1 1 1 1
          + 0 1 0 0 0 0 1
          ---------------------------
            0.0 1 1 0 0 1 0 1 0 0 0 1    N

                  0.1 1 0 1 0    D
                  0.1 0 1 0 1    Q
                + 0 1 1 0 1 0
              + 0 1 1 0 1 0
            + 0 1 0 0 0 0 0 1 0
          + 0 1 1 0 1 0
          ---------------------
            0.1 0 0 0 1 0 0 0 1 0    N
```

where we have utilized the rapid multiplication process as indicated. Comparing these with the previous two cases above, one sees that the shorter examples of divi-

sion are the reverse of rapid multiplication. Thus, during division we desire to perform only operations corresponding to a minimum decomposition of Q. Of course, we don't know Q to begin with, and this is where the problem lies.

In the partial algorithm, the moving of the binary point, *i.e.*, the shifting of N with respect to D, is of course accompanied by a moving of the binary point of Q, since $N=DQ$, and D remains normalized. In the above illustrations we neglected to indicate this, since at the end of the division operation the binary point is replaced in its normal position. However, in the following discussion it is convenient to consider the more correct corresponding binary-point movement in both N and Q, and the discussion of "where the shifting stops" or "should stop" in terms of Q (or $Q^{(s)}$) is to be directly related to the shifting of N (or $N^{(s)}$) with respect to D.

The complete algorithm follows the same flow chart as the partial algorithm (Fig. 2) except for the number of positions α that $N^{(s)}$ is shifted. In the complete algorithm the number of positions shifted may be one less than, equal to, or one more than that required to normalize $N^{(s)}$, depending on rules to be given below. The above bad cases have illustrated examples where shifting one less or one more place is more desirable than normalizing. This occurs because simple normalization will not necessarily result in the best decomposition for Q. If we knew the best decomposition of Q (which as yet we do not), then each successive shift (from left to right) should stop with the binary point *at the right* of a position requiring an operation. For our illustrations, for example, the binary points below indicate the positions at which the successive shifts stopped:

$$Q = \overset{+}{0}.1\bar{1}.000\overset{+}{1}. \qquad Q = 0\overset{+}{1}.0\overset{+}{1}.0\overset{+}{1}.$$

Note that these positions correspond to positions to the right of each successive "?" in the desired rapid division process, where the next following bit of Q after the "?" usually is changed from 0 to 1 or 1 to 0, according to the flow chart of Fig. 2.

Alternative Decompositions

It is important to note that there are cases for which a *choice* of the best decomposition exists, such as in 01011; here $\overset{+}{0}\overset{+}{1}0\bar{1}1$ has as many terms as has $\overset{+}{0}1\bar{0}\bar{1}1$. The same comment applies to 0101011, 010101011, or any number whose normalized magnitude is greater than $2/3(=.10101010\cdots)$ and less than $3/4(=.11)$. In these cases the shift can stop to the right *or* to the left of the leftmost unit, as it is interpreted as an isolated unit or as beginning a string, *e.g.*,

$$0\overset{+}{1}.0\overset{+}{1}.1\bar{1}. \quad \text{or} \quad \overset{+}{0}.1\bar{0}.1\bar{1}.$$

If we are in a string of zeros, moving from left to right, and we come to such a unit, it will be called an *alternative* unit. However, if we are in a string of zeros and come across two or more adjacent units, then the leftmost unit must start a string of units, and the shifting must stop to its left (*i.e.*, to the right of the preceding zero); such a unit is called a *definitive* unit. Analogous definitions hold for alternative and definitive zeros if we are in a string of units.

The rules for determining the best decomposition of Q, working from left to right, are then as follows: If we are in a string of zeros and come to 0100, 010100, etc., then these units must be considered isolated; if we come to a definitive unit, then a string of units is initiated; if we come to an alternative unit, it can be considered either as isolated or as beginning a string of units, depending on the circumstances (see below). If we are in a string of units, analogous rules hold for the first zero encountered.

Modification of the Simplified Algorithm

In the following, we shall deal only with positive $N^{(s)}$, which implies (see the flow chart) that we are in a string of zeros. To see that the discussion also holds for negative $N^{(s)}$, simply note that in such a case $N^{(s)}=D(Q^{(s-1)}-1)$ where $D>0$, and where $(Q^{(s-1)}-1)<0$ contains only the decomposition terms not yet determined by the process; hence multiplying both sides by -1 reduces this to the positive case to be considered.

So far we have been discoursing as if Q were known. The reason for this is that at any point of the process after properly shifting $N^{(s)}$ α positions, the best decomposition of the remaining next few bits of Q, *i.e.*, of the next few bits of $Q^{(s)}$, can be determined by examining $N^{(s)}$ and D. We shall proceed by examining all possible cases for successive bits of $N^{(s)}$ and D.

Consider first $D=.11\cdots$ and $N^{(s)}=0.11\cdots$ (normalized by a shift of, say, δ positions); that is, $0.11\leq D<1$ and $0.11\leq N^{(s)}<1$. Examining the possible range of $Q^{(s)}$ we find $.11/1<Q^{(s)}<1/.11$, or $0.11<Q^{(s)}<1.010101\cdots$ $(=4/3)$. If $0.11<Q^{(s)}<1$, then $Q^{(s)}$ begins with a definitive unit and normalizing has shifted the binary point to the end of the preceding string of zeros, as desired; if $1\leq Q^{(s)}<4/3$, then $Q^{(s)}$ begins with an isolated unit and normalizing has shifted the binary point *past* this isolated unit, as desired. Thus normalizing is proper for all cases where $D=.11\cdots$ and $N=0.11\cdots$.

Next, consider $D=.11\cdots$ and $N^{(s)}=.10\cdots$, as in Case II above. Here the range of $Q^{(s)}$ is $.1/1<Q^{(s)}<.11/.11$, or $0.1<Q^{(s)}<1$. For $2/3\leq Q^{(s)}<1$, the first unit of $Q^{(s)}$ is either alternative or definitive, and normalizing is permissible or required. For $0.1<Q^{(s)}<2/3$, the first unit of $Q^{(s)}$ is isolated but normalizing has not shifted the binary point past this unit as would be

TABLE I*

Cases	D	$N^{(s)}$(norm.)	$Q^{(s)}>$	$Q^{(s)}<$	First Unit	Places Shifted
1	$.110\cdots$	$.101\cdots$	$.\dot{1}0\dot{1}1$	1.0	alt. or def.	δ
2	$.110\cdots$	$.1000\cdots$	$.1\dot{0}0\dot{1}$	.11	iso. or alt.	$\delta+1$
3	$.111\cdots$	$.100\cdots$	.1	$.\dot{1}0\dot{1}1$	iso. or alt.	$\delta+1$
4	$.111\cdots$	$.1011\cdots$	$.\dot{1}011$	$.1\dot{1}0\dot{1}$	alt. or def.	δ
5	$.1100\cdots$	$.1001\cdots$	$.\dot{1}01100010\dot{0}1\dot{1}$	$.1\dot{1}0\dot{1}$	alt. or def.	δ
6	$.1110\cdots$	$.101\cdots$	$.\dot{1}0\dot{1}$	$.1\dot{1}0\dot{1}$	alt. or def.	δ
7	$.1111\cdots$	$.1010\cdots$	$.\dot{1}0\dot{1}$	$.\dot{1}0\dot{1}1$	iso. or alt.	$\delta+1$
8	$.11010\cdots$	$.10010\cdots$	$.\dot{1}0\dot{1}$	$.\dot{1}0\dot{1}1101100010\dot{0}$	alt.	either
9	$.11010\cdots$	$.10011\cdots$	$.\dot{1}0\dot{1}10011\dot{1}\dot{1}\dot{0}$	$.\dot{1}10001001\dot{1}\dot{0}$	alt. or def.	δ
10	$.11011\cdots$	$.10010\cdots$	$.\dot{1}0\dot{1}00\dot{1}$	$.\dot{1}0\dot{1}1\dot{0}1000010010\dot{1}\dot{1}$	iso. or alt.	$\delta+1$
11	$.11011\cdots$	$.10011\cdots$	$.\dot{1}0\dot{1}0\dot{1}$	$.\dot{1}0\dot{1}1\dot{1}1011010000\dot{1}0\dot{0}$	alt.	either

* The superscript dots mark the beginnings and ends of repeating fractions; thus, for example, $.\dot{1}0\dot{1}\dot{1}=.1011011011011011\cdots$.

desired; hence for this range of $Q^{(s)}$ we would have to shift one more place, *i.e.*, $\alpha=\delta+1$ positions. To find those cases in which the values of $N^{(s)}$ and D require shifting $\delta+1$ positions, without having to determine if $0.1D<N^{(s)}<2D/3$, we analyze succeeding bit positions of $N^{(s)}$ and D, making use of the fact that if $2/3<Q^{(s)}<1$ then the first unit is alternative and may be passed or not in shifting, as is convenient.

For example, consideration of three bits each of $N^{(s)}$ and D give four cases:

a) $D=.110\cdots$, $N^{(s)}$ (normalized) $=.101\cdots$;
b) $D=.110\cdots$, $N^{(s)}$ (normalized) $=.100\cdots$;
c) $D=.111\cdots$, $N^{(s)}$ (normalized) $=.101\cdots$;
d) $D=.111\cdots$, $N^{(s)}$ (normalized) $=.100\cdots$.

For the first of these cases,

$$.101/.111 < Q^{(s)} < .11/.11$$

or

$$0.1011011011011\cdots < Q^{(s)} < 1.$$

Hence for this case the first unit of $Q^{(s)}$ is either alternative or definitive, and normalizing, by shifting δ places, is permissible or required. However, for the fourth case,

$$.100/1 < Q^{(s)} < .101/.111$$

or

$$0.1 < Q^{(s)} < 0.1011011011011\cdots.$$

For this case, if $0.1<Q^{(s)}<2/3$, then the first unit of $Q^{(s)}$ is isolated, and we should shift one place further, *i.e.*, $\delta+1$ places in all. Also, if

$$2/3 \leq Q^{(s)} < 0.1011011011\cdots,$$

then the first unit of $Q^{(s)}$ is alternative, so that shifting $\delta+1$ places is required or permissible over the entire range of $Q^{(s)}$ for this case. On the other hand, the range of $Q^{(s)}$ for the second and third cases is such that the first unit of $Q^{(s)}$ may still be isolated, alternative, or definitive; for these two cases, the ranges of $Q^{(s)}$ must be computed for all possibilities of the fourth bits (and in a few of those cases, the fifth bits) of $N^{(s)}$ and D. Table I summarizes the definitive results of such computations.

From Table I, cases 2, 3, 7, and 10 require shifting $N^{(s)}$ one place farther ($\alpha=\delta+1$ places in all) because of the isolated units; Cases 8 and 11 *allow* shifting farther, if it is desirable. Simplification by Boolean Algebra gives the following rule (e indicates the bit may be either zero or one):

Rule I: If $D=.11\cdots$, and if $N^{(s)}$ normalized by shifting δ places would equal $.10\cdots$, then in the flow chart of Fig. 2, shift $\alpha=\delta+1$ positions

1) if D begins $.111ee$ and $N^{(s)}$ begins $100ee$
2) if D begins $.11eee$ and $N^{(s)}$ begins $1000e$
3) if D begins $.1111e$ and $N^{(s)}$ begins $10e0e$
4) if D begins $.11e11$ and $N^{(s)}$ begins $100ee$

or alternatively

4) if D begins $.11e1e$ and $N^{(s)}$ begins $100e0$;

otherwise shift $\alpha=\delta$ positions.

Consider now $D=.10\cdots$ when $N^{(s)}$ (normalized by shifting δ positions) $=.10\cdots$. The range of $Q^{(s)}$ is $2/3<Q^{(s)}<1.1(=3/2)$. If $2/3<Q^{(s)}<1$, the first unit is either alternative or definitive and normalizing is correct; if $1\leq Q^{(s)}<1.1$, the first unit is an isolated (or alternative) unit which has properly been passed ($N^{(s+1)}>0$ generates this unit on the next differencing).

For $D=.10\cdots$ and $N^{(s)}$ (normalized) $=.11\cdots$ (as in Case I above, for example), the range of $Q^{(s)}$ is given by $.11/.11<Q^{(s)}<1/.10$ or $1<Q^{(s)}<10(=2)$. For $1<Q^{(s)}<4/3$, the first unit of $Q^{(s)}$ is isolated; normalizing has shifted the binary point past this first unit, as desired. For $4/3\leq Q^{(s)}<3/2$ (*i.e.*, $1.01010101\cdots \leq Q^{(s)}<1.1$), the first unit of $Q^{(s)}$ is alternative, and normalizing would be permissible. However, for $3/2\leq Q^{(s)}<2$, the first unit of $Q^{(s)}$ is definitive, but normalizing has shifted the binary point beyond it; for $Q^{(s)}$ on such range, we must shift one less place, *i.e.*, $\alpha=\delta-1$ places. An analysis of the range of $Q^{(s)}$ for succeeding bits of $N^{(s)}$ and D, similar to that above, gives the following rule:

Rule II: If $D=.10\cdots$, and if $N^{(s)}$ normalized by shifting δ places would equal $.11\cdots$, then in the flow chart of Fig. 2, shift $\alpha=\delta-1$ positions

1) if D begins $.100ee$ and $N^{(s)}$ begins $111ee$

2) if D begins .1000e and $N^{(s)}$ begins 11eee
3) if D begins .10e0e and $N^{(s)}$ begins 1111e
4) if D begins .100ee and $N^{(s)}$ begins 11e11

or alternatively

4) if D begins .100e0 and $N^{(s)}$ begins 11e1e;

otherwise shift $\alpha=\delta$ positions.

Rules I and II, with the flow chart of Fig. 2, constitute the complete algorithm.

Justification of Rule II

Note that the exceptional cases of Rule II are the same as those of Rule I with the bits for $N^{(s)}$ and D interchanged. That this is to be expected, and in fact, that it justifies the second rule when the first is proved, is shown by the following considerations (which have been mentioned above). With $N^{(s)}$ and D both normalized the possible range for $Q^{(s)}$ is

$$1, 2 < Q^{(s)} < 2.$$

The range for which normalizing is required is

$$3/4 \leq Q^{(s)} \leq 4/3$$

whose limits are reciprocals. The range for which shifting δ or $\delta-1$ places is optional is

$$4/3 < Q^{(s)} < 3/2,$$

and that for which shifting δ or $\delta+1$ is optional is

$$3/4 > Q^{(s)} > 2/3.$$

The respective limits of these two ranges are again reciprocal. And finally, the limits of the range for which shifting $\delta-1$ positions is required, *i.e.*,

$$3/2 \leq Q^{(s)} < 2,$$

are respectively the reciprocals of the limits of the range for which shifting $\delta+1$ positions is required, *i.e.*,

$$2/3 \geq Q^{(s)} > 1/2.$$

Hence the range for the first rule for shifting $\delta+1$ places is the reciprocal of the range for the second rule for shifting $\delta-1$ places, thus interchanging $N^{(s)}$ and D.

Justification of Alternative Decomposition

In order to prove that the division procedure described herein requires the fewest possible additions and subtractions, we make use of a "canonical" decomposition, which has the property that addition and/or subtraction is never required for two consecutive binary positions. Both Tocher[4] and Reitwiesner[5] have proved that no other decomposition can be found that requires fewer total additions and subtractions; in fact, Reitwiesner proves the uniqueness of the canonical decomposition. Based on this result, the essence of our proof requires merely that we show that when our alternative decompositions do require additions and/or subtractions for two consecutive binary positions, then the decomposition can be rearranged to form a canonical decomposition with the same total number of additions and subtractions. To see this, we first note that there are only sixteen conceivable ways of having two adjacent operations, and only two of these ways can actually occur in our decomposition. For example, $\cdots \overset{++}{11} \cdots$, $\cdots \overset{+-}{11} \cdots$, etc., cannot occur; only $\cdots \overset{++}{10} \cdots$ and $\cdots 0\bar{1} \cdots$ can occur. For instance, we might have

$$\cdots 00\overset{+}{1}0\overset{++}{1}011\bar{1}00 \cdots \quad \text{or} \quad \cdots 0\overset{+}{0}111\bar{0}\bar{1}0100 \cdots .$$

These cases may be rearranged into canonical form as

$$\cdots 00\overset{+}{1}0\bar{1}011\bar{1}00 \cdots \quad \text{and} \quad \cdots 0\overset{+}{0}111\bar{0}\overset{+}{1}0\overset{+}{0}100 \cdots$$

respectively, with no change in the number of operations. These observations can be generalized.

Rapid Division in Computers

The Remainder

In nonrestoring division, the current remainder is most commonly retained in complemented form when overdraft occurs. Our tables express the shifting criteria in terms of the leading bits of the absolute values of the divisor and current remainder. In practical implementation, these tables would be modified, in accordance with the design of a particular machine, to accommodate a complemented current remainder.

On a computer, the division process must end when the last allotted place of Q has been filled. If this occurs on a differencing with the resulting $N^{(s+1)}$ positive, then from (8) and (12) or (15) and (18), this $N^{(s+1)}$ is the final remainder; if it occurs on a differencing with $N^{(s+1)}$ negative, then from (8) and (10) or (15) and (17) the remainder is $N^{(s+1)}+D=D-\left|N^{(s+1)}\right|$. If the last place of Q is filled on a shift, shifting ceases: if $N^{(s+1)}$ is positive, the shifted $N^{(s+1)}$ is the remainder; if $N^{(s+1)}$ is negative, the remainder is D plus the shifted $N^{(s+1)}$.

Serial vs Parallel Computers

A serial computer may be constructed to use the algorithm provided that a shift-register, rather than a delay-line, accumulator is used. Differencing and normalizing would occur in the same recirculation time, the latter simply by appropriately delaying the entry of $N^{(s)}$ into the adder-subtractor. The number of zeros on the most significant end, and the most significant bits, of $N^{(s)}$ can be recorded in a zero-bit counter, and a set of flip-flops, all reset by a new unit in the result.

For a parallel computer, time can be saved only by shifting and differencing both in the same time interval. The statistical analysis that Smith and Weinberger[2] have made for the effectiveness of shortcut multiplication is based on the probable occurrence of various lengths of strings in the multiplier; the same probabilities, and hence the same time savings (with regard to numbers of differencings), apply to the quotient. Fig. 3 summarizes their results, as they apply to division.

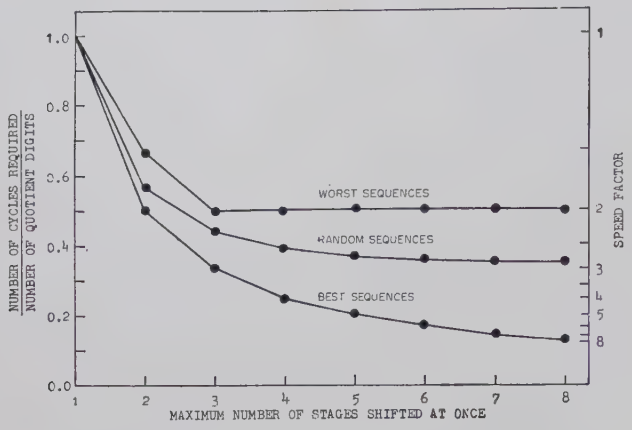


Fig. 3—Effectiveness of the complete algorithm for rapid division.

From the figure, it is evident, as Smith and Weinberger have concluded, that for random sequences or worse the number of cycles saved increases only slightly for more than 4 places for the maximum shift. For unlimited shifting, as in a serial computer, the ratio of required cycles to quotient bits approaches one-third asymptotically, so that when applied to a serial computer the complete algorithm will result, on the average, in a saving of two-thirds in division time.

Acknowledgment

The authors wish to express their appreciation to G. W. Reitwiesner for several valuable suggestions on this paper.

A General Junction-Transistor Equivalent Circuit for Use in Large-Signal Switching Analysis*

S. B. Geller†, P. A. Mantek†, member, ire, and D. R. Boyle†, member, ire

Summary—**This paper describes a general medium-speed junction-transistor equivalent circuit that is valid for large- and small-signal operation. Solutions of nonlinear differential equations written from the equivalent circuit for its response to large-signal driving pulses are obtained by graphical, analytic, and analog computer techniques. The static base-to-emitter *V-I* characteristic is used extensively in the various analysis techniques and produces the proper response to a driving source of any internal resistance.**

In Section II a charge-control equation is employed for constructing the equivalent circuit in a common-emitter hybrid π configuration. Section III deals with the techniques for solving large-signal switching problems. Section IV discusses some of the required measurements.

I. Introduction

THE difficulties that are encountered in the analysis of junction-transistor large-signal switching circuits are compounded when the driving source impedances are so low as to permit the input characteristics of the transistor to influence the driving current. These difficulties are usually avoided by postulating constant-current driving conditions. This is an unrealistic assumption for many practical applications. It is proposed that the nonlinear base-to-emitter characteristics of the junction transistor may be employed advantageously in solving various types of large-signal switching problems, regardless of the driving source impedance.

The problem is attacked with a medium-speed junction-transistor equivalent circuit that is developed from a basic charge-control equation. Its current-sensitive base-to-emitter capacitance is evaluated in the time domain and includes the dynamic collector-to-base feedback effects. The static base-to-emitter *V-I* characteristic is employed in conjunction with an invariant time-constant factor in order to obtain the steady-state and transient solutions of a nonlinear differential equation that is written for the input network. This is accomplished by graphical, analytic, and analog computer techniques that are discussed in detail. The circuit also incorporates saturation and storage-time simulation facilities, and is applicable for all modes of typical transistor operation, *i.e.*, both small or large signal applications.

Figs. 1 and 2 display some analog-simulated results that were obtained from the junction-transistor general equivalent circuit. The simulated waveforms have been placed above the corresponding oscilloscope records, which were taken from an actual 2N414 alloy junction transistor under identical operating conditions. The figures show the response of a common-emitter resistance-loaded amplifier driven by a rectangular pulse.

* Received by the PGEC, January 28, 1960; revised manuscript received, February 24, 1961. This work was sponsored by the Air Force Cambridge Research Center, Bedford, Mass., the National Bureau of Standards, and the Office of the Chief Signal Officer, U. S. Army.

† National Bureau of Standards, Washington, D. C.

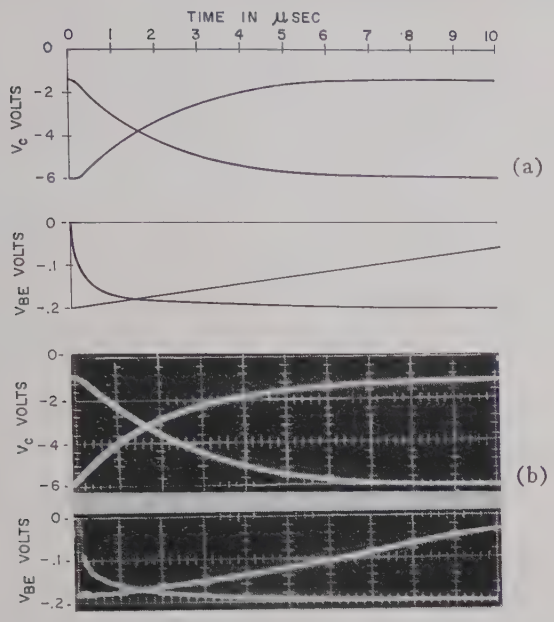


Fig. 1—Analog simulation of switching cycle (constant-current drive). (a) Simulated. (b) Experimental.

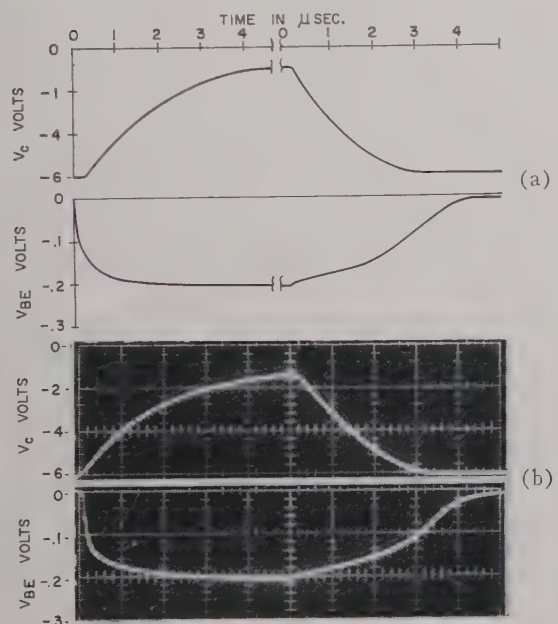


Fig. 2—Analog simulation of switching cycle (source resistance =3K ohms). (a) Simulated. (b) Experimental.

Fig. 1 displays the approximately equal rise and fall times of the collector voltage v_C as is typical for a transistor driven from a constant-current source. The simulated waveforms of base voltage v_{BE} display the same steep rise and linear decay characteristics that appear in the actual oscilloscope presentation. Fig. 2 displays the response to a more practical situation in which the source impedance is 3K ohms. The collector voltage rise time is now greater than its fall time, when the transistor is driven from cutoff to near saturation. The simulated results shown in Fig. 2 display this collector rise-to-fall time relationship with the proper magnitudes. The simulated base voltage v_{BE} waveforms have the same steep rise and humpbacked decay characteristics that appear in the actual oscilloscope presentation.

II. Description of Equivalent Circuit

Beaufoy and Sparkes[1] have shown that the time dependency of the stored base-charge components in the junction transistor can be obtained from the solution of

$$I = \frac{q_B}{T_B} + \frac{dq_B}{dt} + \frac{dq_{BN}}{dt}, \tag{1}$$

where I is a base-current step.[2]

A possible base-to-emitter network represented by (1) is shown in Fig. 3. The base-to-emitter diffusion capacitance $c_{b'e}$ and the feedback capacitance c_f are defined as (see Appendix)

$$c_{b'e} = 39 i_{B'E} T_B, \tag{2}$$

$$c_f = 39 i_{B'E} R_L C_c \alpha_{fe}, \tag{3}$$

where $i_{B'E}$ is the total instantaneous value of the resistive base current component. The number 39 is used symbolically to represent q/kT in this paper ($q/kT = 39$ volts^{-1} at room temperature).

The nonlinear resistance R' is defined by the $V_{B'E}$ vs $I_{B'E}$ base-to-emitter internal junction characteristic. Fig. 4 displays the characteristic for several 2N414 alloy junction transistors. The instantaneous static value of R' at each point on the curve is

$$r_{B'E} = \frac{1}{39 i_{B'E}} \ln \frac{(\alpha_{FE} + 1) i_{B'E}}{I_{ES}}. \tag{4}$$

The dynamic value of R' seen by the driving source during transient intervals is

$$r_{b'e} = (39 i_{B'E})^{-1}. \tag{5}$$

This dual character of R' produces proper dynamic and quiescent network responses.

By incorporating the dq_{BN}/dt current into the base-to-emitter network, the feedback effects of the collector-to-base space charge capacitance C_c are directly contained in the solution of $i_{B'E}$ vs t when a finite collector load resistance R_L exists. The expression for $i_{B'E}$ vs t is the solution of

$$I = i_{B'E} + (T_B + R_L C_c \alpha_{fe}) \frac{di_{B'E}}{dt}. \tag{6}$$

Once $i_{B'E}(t)$ is known, the total instantaneous collector current i_C is determined; *i.e.*,

$$i_C(t) = \alpha_{FE} i_{B'E}(t) = \alpha_{FE} \frac{v_{B'E}(t)}{r_{B'E}(t)}. \tag{7}$$

[1] R. Beaufoy and J. J. Sparkes, "The junction transistor as a charge-controlled device," *ATE J.*, vol. 13, pp. 310–327; October, 1957.

[2] The other symbols are defined in a list at the end of this paper.

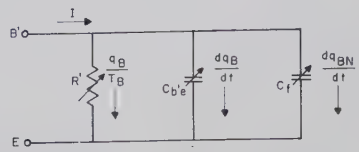


Fig. 3—Network representation for (1).

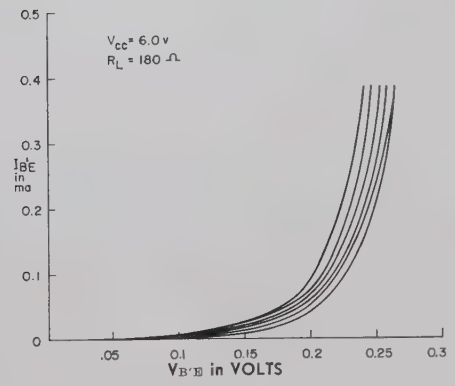


Fig. 4—$V_{B'E}$ vs $I_{B'E}$ curves for 2N414 junction transistors.

When the junction transistor is switched by a large base-current step but does not saturate, $i_C(t)$ is approximately an exponential function.[3,4] This type of response is simulated by the nonlinear network of Fig. 5, when T_B' (where, $T_B' = T_B + R_L C_c \alpha_{fe}$) is considered to be constant over the entire base-current excursion. The network then displays a single pole response and has the invariant time constant

$$r_{b'e}(c_{b'e} + c_f) = (39 i_{B'E} T_B')(39 i_{B'E})^{-1} = T_B'. \quad (8)$$

The solution of (6) with T_B' = constant is

$$i_{B'E}(t) = I\left[1 - \exp\left(-\frac{t}{T_B'}\right)\right]. \quad (9)$$

Although the nonlinear network elements will be constrained to maintain an invariant T_B', the dynamic impedance seen looking into the B'–E terminals is continuously variable during the transient interval. This variation is important when source impedances are of low values, since then the input impedance affects the level of current from the driving source. The variations in the rise (and fall) times between the simulated curves of Figs. 1 and 2 show the effect of dynamic impedance.

Fig. 6(a) displays the complete general equivalent circuit in a hybrid π common-emitter configuration. The extrinsic resistances r_{bb}' and r_s, the collector current generator $\alpha_{FE} i_{B'E}$, and the I_{CB0} leakage current generator complete the active region portion of the equivalent circuit. The base-to-emitter capacitance C is equal to $c_{b'e} + c_f$. The components enclosed by the dashed line are

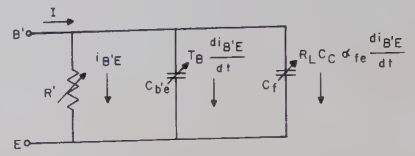
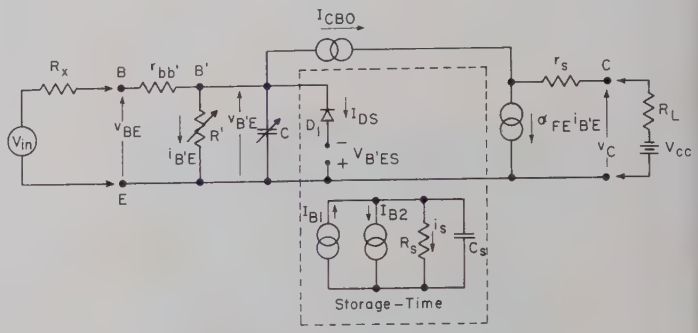
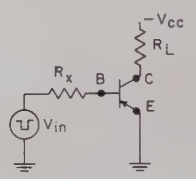


Fig. 5—Network with branch currents described in (6).



(a) Complete medium-speed general equivalent circuit in common-emitter configuration.



(b) Common-emitter resistance-loaded amplifier stage.

Fig. 6.

the saturation sensing diode D_1 and the storage time circuit.

A. Ideal Diode D_1

The V_{BE} vs $I_{B'E}$ external junction plot for a resistance-loaded common-emitter stage will display a breakpoint when $V_{BE} = V_{B'ES} + I_{B'ES} r_{bb}'$ (Fig. 7). The breakpoint occurs at the onset of saturation when the sudden limiting of the collector current is accompanied by an abrupt decrease in terminal impedances. The biased ideal diode D_1 in Fig. 6(a) enables the equivalent circuit to simulate these effects. The placement of D_1 across the B'–E terminals can be justified from the back-to-back diode representation of the junction transistor[5] if the collector-to-emitter voltage is considered to be clamped to a final level in saturation.

D_1 is assumed to be biased with

$$V_{B'ES} = \frac{V_{CC} R_{B'ES}}{(R_L + r_s)\alpha_{FES}}. \quad (10)$$

When $v_{B'E}$ equals $V_{B'ES}$ at saturation, D_1 switches on and clamps the B'–E junction to the voltage $V_{B'ES}$. The collector terminal current is therefore clamped to

[3] J. J. Ebers and J. L. Moll, "Large-signal behavior of junction transistors," PROC. IRE, vol. 42, pp. 1761–1772; December, 1954.

[4] J. W. Easley, "The effect of collector capacity on the transient response of junction transistors," IRE TRANS. ON ELECTRON DEVICES, vol. ED-4, pp. 6–14; January, 1957.

[5] R. F. Shea, "Principles of Transistor Circuits," John Wiley and Sons, Inc., New York, N. Y., pp. 403–407; 1953.

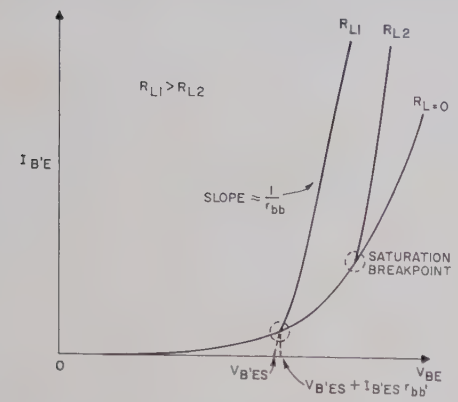


Fig. 7—V_{BE} vs $I_{B'E}$ plot showing breakpoint at saturation.

the saturated value of

$$I_{CS} = \frac{V_{CC}}{R_L + r_s} = \frac{V_{B'ES}}{R_{B'ES}} \alpha_{FES}. \tag{11}$$

Any further increase in base driving current beyond $I_{B'ES}$ flows only through D_1 and the extrinsic base resistance $r_{bb'}$. This excess driving current I_{DS} through D_1 produces the storage time effects.

The collector circuit, which acts like an $\alpha_{FE} i_{B'E}$ current source in the active region, looks like a low-impedance source in saturation because the load resistance R_L influences the diode bias level $V_{B'ES}$. This permits R_L to control the final collector-current levels. Increased base drive into saturation reduces the collector-current rise time but has no effect on its decay time. This action is simulated due to D_1 because the decay cycle is always initiated from the same $V_{B'ES}$ level when D_1 switches "OFF" regardless of the turn-on overdrive applied.

B. Storage Circuit

At the onset of collector saturation the diode current I_{DS} is accompanied by the storing of excess charge Q_{BS} in the base region of the transistor. The ratio of the charge to the current is defined as

$$K_s = \frac{Q_{BS}}{I_{DS}}. \tag{12}$$

The following storage time equation[1,2,6] includes the effects of the turn-on current step I_{B1} and the positive or negative base current at turn-off I_{Br}:

$$t_s = K_s \ln \frac{I_{B1} - I_{B2}}{I_{B'ES} - I_{B2}} \text{ (storage time)}. \tag{13}$$

K_s is measured with the base current at the approximate level expected during operation. A measurement technique for K_s is treated in detail by Simmons.[6]

It has been shown that the equivalent circuit response agrees with Moll's[7] assumption that $r_{bb'}$ is the only significant base-to-emitter impedance seen by the driving source in saturation. In order to simulate the time-delay effects that are associated with carrier storage, an adjunct *storage-time* network is employed (see Fig. 6). It is constructed by analogy to (13) with $K_s = R_s C_s$ and the driving currents represented by the I_{B1}' and I_{B2} current generators. I_{B1}' is equal to I_{B1} and begins to flow when I_{B1} switches off. The time required for the current through R_s to attain any predetermined value is

$$t = R_s C_s \ln \frac{I_{B1}' - I_{B2}}{(I_{B1}' - i_s) - I_{B2}} \tag{14}$$

where i_s is the current in R_s. The time t in (14) is equal to the storage time t_s when $I_{B1}' - i_s = I_{B'ES}$. This occurs when $i_s = I_{DS}$ because $I_{B1} = I_{DS} + I_{B'ES}$. Hence, the final current I_{DS} through D_1 may be used as a reference current to indicate the terminating instant of storage time when D_1 is switched "OFF" and the decay interval begins. It is assumed here that equilibrium conditions exist prior to the storage interval.

C. I_{CB0}

The I_{CB_0} current generator, where I_{CB_0} is defined as the reverse saturation current of the collector junction with zero emitter current, has been included in the general equivalent circuit because of its effects on transistor circuit stability. The component of collector current that flows due to I_{CB_0} is

$$i_C = \left(1 + \frac{(R_x + r_{bb'})\alpha_{FE}}{(R_x + r_{bb'}) + r_{B'E}}\right) I_{CB0}. \tag{15}$$

III. Large Signal Switching Analysis

The common-emitter resistance-loaded amplifier is the most prevalent building block used in switching circuit applications. Therefore, its large signal response will be investigated by graphical and analytic techniques. The objective in each method is to solve the nonlinear differential equation

$$(R_x + r_{bb'}) \left[T_B' \frac{di_{B'E}}{dt} + i_{B'E} \right] + v_{B'E} = V_{\text{in}}, \tag{16}$$

which is written for the base-to-emitter network of Fig. 6(a).

A. Graphical Techniques

Eq. 17 expresses $di_{B'E}/dt$ at every instant for the basic common-emitter stage [Fig. 6(b)] when driven by a rectangular voltage pulse V_{in}:

$$\frac{di_{B'E}}{dt} = \left[\frac{\frac{V_{\text{in}}}{R} - \frac{v_{B'E}}{R} - i_{B'E}}{T_B'} \right], \tag{17}$$

where $R = R_x + r_{bb'}$ and $T_B' = \text{constant}$.

This nonlinear differential equation provides basis for the graphical solution which plots $i_{B'E}$ vs time.

[6] C. D. Simmons, "Hole storage delay time and its prediction," Lansdale Tube Co., Lansdale, Pa., Application Rept. No. 234; 1957.
[7] J. L. Moll, "Large signal transient response of junction transistors," Proc. IRE, vol. 42, pp. 1773–1784; December, 1954.

B. Graphical Solution

Eq. 17 is approximated over a small interval j in incremental form:

$$\frac{(\Delta i_{B'E})_j}{(\Delta t)_j} = \frac{\dfrac{V_{in}}{R} - \dfrac{(v_{B'E})_j}{R} - (i_{B'E})_j}{T_B'}. \tag{18}$$

Then, if either the current or time is partitioned into appropriate intervals, the value of $i_{B'E}$ and t can be found from

$$(i_{B'E})_{j+1} = (i_{B'E})_j + (\Delta i_{B'E})_j, \tag{19}$$

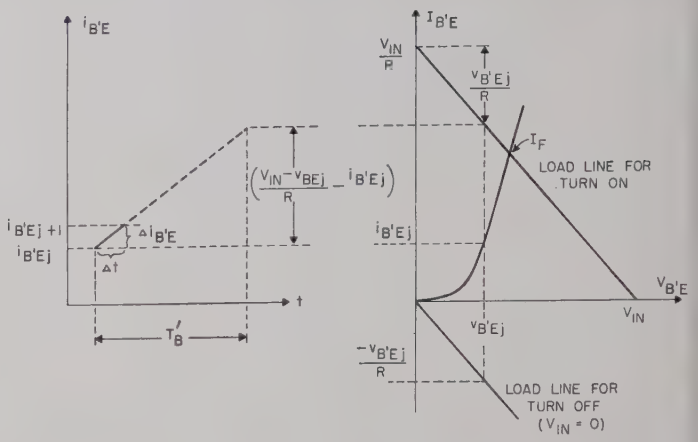
$$(t)_{j+1} = (t)_j + (\Delta t)_j. \tag{20}$$

The graphical interpretation of (18–20) is shown in Fig. 8(a) for a single rise-time interval. In the examples of Fig. 8(b) and 8(c) the current axis has been partitioned into three equal intervals. Fig. 8(b) demonstrates the construction for the turn-on transient and Fig. 8(c) for the turn-off transient. The construction starts at point A which is the initial value of $i_{B'E}$ for the first interval. The horizontal projection of this point onto the $V_{B'E}$-$I_{B'E}$ characteristic gives the initial value of $v_{B'E}$ at point B. Point B is projected vertically until it intersects the load line at point C. The distance B–C is equal to the numerator of (18). The distance A–D is constructed equal to T_B'. This makes the slope of line A–E equal to $\Delta i_{B'E}/\Delta t$ for the first interval. Point F is the final condition for the first interval and the initial condition for the second interval. The above argument also applied to the second and third intervals provided that the points A, B, C, D, E, F be replaced in the discussion by F, G, H, I, J, K, respectively, for the second interval and K, L, M, N, O, P, respectively, for the third interval.

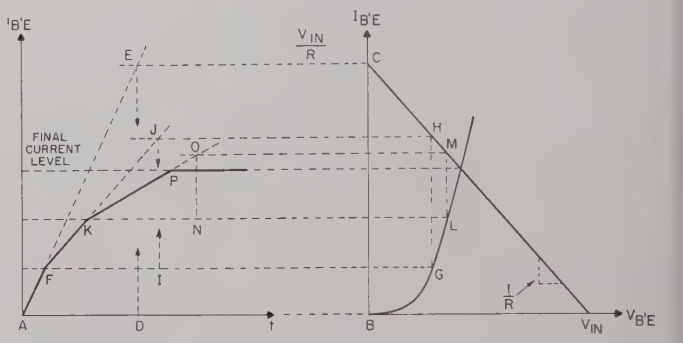
The construction for the turn-off transient is a solution of (18) for $V_{in}=0$. This is represented in the construction of Fig. 8(c) by drawing the load line of R so that it intersects the origin from below the $V_{B'E}$ axis. The construction is initiated from the $i_{B'E}$ current level that existed just prior to the beginning of the turn-off transient. Figs. 9 and 10 show the construction using six intervals for cases in which the driving source resistance is 3K and infinite, respectively.

The collector current i_C is assumed to be the result of $i_{B'E}$ multiplied by the chosen current gain factor α_{FE}, where α_{FE} is evaluated at the final collector current level. This can be displayed in the construction by an additional axis that is parallel to the $i_{B'E}$ axis but whose ordinate values differ by the constant factor α_{FE}. If the effects of varying current gain are desired in the construction, the ordinates of the i_C axis may be evaluated according to the effective current gain in each interval.

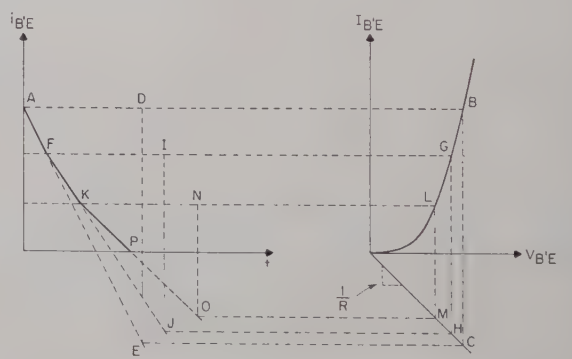
For the case in which the transistor is driven into the saturated region, the iteration process is concluded when $i_{B'E}=I_{B'ES}$; hence, the collector signal current during a switching cycle will be in the range



(a) Graphical interpretation of (18)–(20).



(b) Graphical construction method for a turn-on transient.



(c) Graphical construction method for a turn-off transient.

Fig. 8.

$\alpha_{FE}i_{B'E} \leq i_C \leq \alpha_{FES}I_{B'ES}$. The steady-state value of base current I_{B1} is determined by the intersection of R and $V_{B'E}$ vs $I_{B'E}$ curve and is shown in Fig. 11 to be

$$I_{B1} = I_{B'ES} + I_{DS}. \tag{21}$$

In the graphical procedure a $V_{B'E}$ vs $I_{B'E}$ curve is plotted with the lowest anticipated value of load resistance R_L placed in the transistor collector circuit. This same plot may then be employed in conjunction with higher value of R_L whose associated breakpoints can be constructed on the curve at points calculated by

$$I_{B'ES} = \frac{V_{CC}}{(R_L + r_s)\alpha_{FES}}. \tag{22}$$

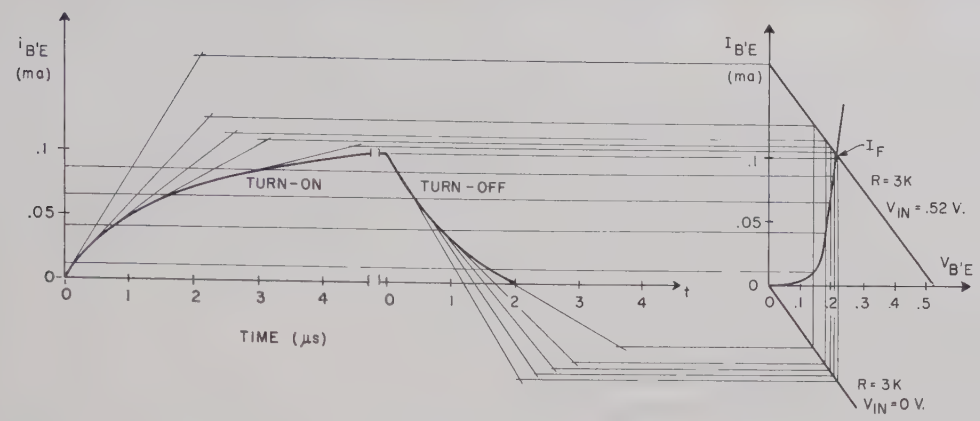


Fig. 9—Graphical analysis of transistor pulse response when driven from a source with 3K ohms resistance.

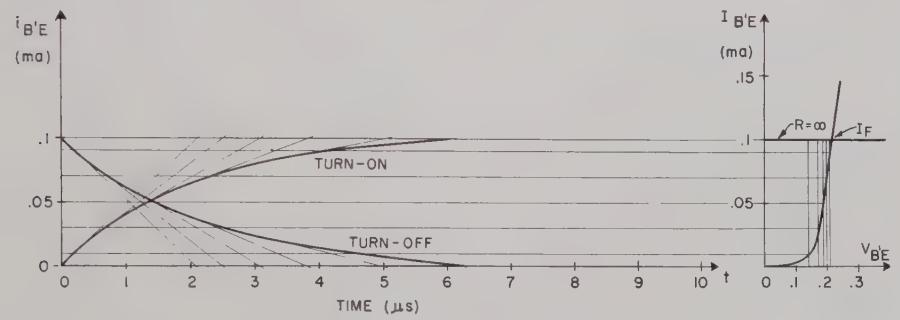


Fig. 10—Graphical analysis of transistor pulse response when driven from a constant-current source.

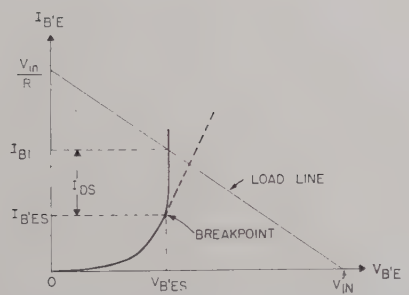


Fig. 11—$V_{B'E}$ vs $I_{B'E}$ curve for use in graphical analysis that extends into saturated region.

This graphical technique leads to insight into the large-signal pulse response of the junction transistor. The slope of the load line of R which is superimposed on the $V_{B'E}$ vs $I_{B'E}$ curve varies from its horizontal position in the constant-current case ($R \rightarrow \infty$) towards a vertical position as $R \rightarrow r_{bb'}$. In the constant-current case all of the $i_{B'E}$ vs t segments are drawn to the same final current level (Fig. 10). The nonlinearities of the input network are ineffective in this case, and only the final level of $i_{B'E}$ that exists at the intersection of R and the $V_{B'E}$ vs $I_{B'E}$ curve and the time constant T_B' are required for the construction process. This explains why the assumption of a constant-current source simplifies the analysis of large-signal switching operation. As the value of R decreases (Fig. 9), the final current levels in the construction vary in each interval; hence, the nonlinear input network played an important role in the development of the transient. This is the reason that the turn-off time is less than the turn-on time when the transistor is operated in the large-signal nonsaturating mode. An application of this graphical technique appears in Ledley.[8]

C. Analytic Solution

The nonlinear differential equation (16) can be solved analytically for collector-current rise and fall times by replacing the current variable $i_{B'E}$ by properly weighted $v_{B'E}$ terms

$$i_{B'E} = Av_{B'E} + Bv_{B'E}^2, \tag{23}$$

where A and B are constants that are evaluated to fit (23) to the $V_{B'E}$ vs $I_{B'E}$ curve at the $0.1I_F$ and $0.9I_F$ current levels. I_F is the final base-current level that is located at the intersection of the curve with the superimposed load line consisting of $R = r_{bb'} + R_x$ (Figs. 9 and 10). The substitution for $i_{B'E}$ in (16) yields the following integrals:

[8] R. S. Ledley, "Digital Computer and Control Engineering," McGraw-Hill Book Co., Inc., New York, N. Y.; 1960.

$$t_2 - t_1 = T_B' \left\{ A \int_{v(0.1I_F)}^{v(0.9I_F)} \frac{dv_{B'E}}{\frac{V_{in}}{R} - \left[\left(\frac{1}{R} + A\right)v_{B'E} + Bv_{B'E}^2\right]} + 2B \int_{v(0.1I_F)}^{v(0.9I_F)} \frac{v_{B'E}\,dv_{B'E}}{\frac{V_{in}}{R} - \left[\left(\frac{1}{R} + A\right)v_{B'E} + Bv_{B'E}^2\right]} \right\}. \quad (24)$$

The voltage limits in the integrals are located directly at 10 per cent and 90 per cent of final base-current points of $V_{B'E}$ vs $I_{B'E}$ curve. In the saturated case the breakpoint on the curve at $I_{B'ES}$ defines the final current level (Fig. 11).

The solution of (24) for the 10–90 per cent collector-current rise time t_r is

$$t_r = T_B' \left[\ln \frac{I_1}{I_9} + \frac{1}{\sqrt{N}} \ln \frac{(P_1 + \sqrt{N})(P_9 - \sqrt{N})}{(P_9 + \sqrt{N})(P_1 - \sqrt{N})} \right], \quad (25)$$

where

$$I = \frac{V_{in}}{R} - \left[\left(\frac{1}{R} + A\right)v_{B'E} + Bv_{B'E}^2\right]$$

$$N = 4V_{in}BR + (1 + AR)^2$$

$$P = 1 + (2Bv_{B'E} + A)R.$$

The subscripts 1 and 9 in (25) indicate that $v_{B'E}$ is set to the limits $v_{B'E}=v(0.1I_F)$ and $v_{B'E}=v(0.9I_F)$, respectively, in P and I.

The limits are reversed for fall-time computations. In the case of constant current drive set $V_{in}/R = I_F$.

D. Analog Computer Simulation

The response of the common-emitter amplifier in Fig. 12 to a rectangular driving pulse applied through

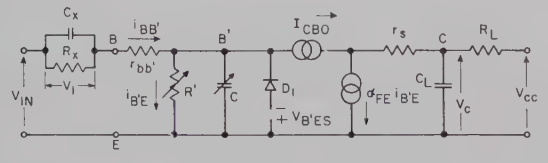


Fig. 12—Common-emitter amplifier circuit simulated on analog computer.

an RC network has been simulated with an analog computer. The dynamics of the circuit during the turn-on and turn-off transients is determined by the interaction of the driving source impedance with the base input network as shown by the governing differential equations (26)–(28):

$$\frac{dv_1}{dt} = \frac{1}{C_X}\left[i_{BB'} - \frac{v_1}{R_x}\right], \quad (26)$$

$$\frac{di_{B'E}}{dt} = \frac{1}{T_B'}\left[i_{BB'} + i_{CB0} - i_{B'E}\right], \quad (27)$$

$$\frac{dv_C}{dt} = \frac{1}{C_L}\left[\frac{(V_{CC} - v_C)}{R_L} - \alpha_{FE} i_{B'E}\right], \quad (28)$$

where

$$i_{BB'} = \frac{1}{r_{bb}}\left[V_{in} - v_1 - v_{B'E}\right].$$

$v_{B'E}=v_{B'E}(i_{B'E})$ and is obtained directly from a curve-following function generator. The simulation will handle correctly situations in which

$$0 \leq R_x \leq \infty.$$

The transistor enters saturation when

$$i_{B'E} = I_{B'ES}.$$

The storage cycle begins at the instant that the input signal decreases below the value required to maintain saturation. The storage period lasts until

$$i_s = I_{B1}' - I_{B'ES}.$$

The current remains clamped at $I_{B'ES}$ throughout the storage period. At every instant during the storage interval, i_s is rising toward $I_{B1}'+i_{BB'}$ with the time constant K_S. Thus,

$$\frac{di_s}{dt} = \frac{1}{K_s}\left[I_{B1}' + i_{BB'} - i_s\right]. \quad (29)$$

It is this feature of the storage circuit that gives the storage time its proper dependence upon driving impedance. This is demonstrated in Table I which compares the simulated with experimental storage time.

TABLE I

STORAGE TIME (μSEC)

V_{in} (volts)	C_X ($\mu\mu$f)	Simulated (K_S=2.05 μsec)	Experimental
1.0	5	0.8	0.6
1.5	5	1.0	1.0
2.0	5	1.2	1.5
1.5	150	0.4	0.5

It is encouraging to note that, although K_S is current sensitive, a fixed value was used to obtain the above results.

At the end of the storage interval $i_{B'ES}$ is unclamped and the turn-off transient begins. Figs. 1 and 2 show some analog simulated results obtained from (26)–(28).

E. Small Signal Mode of Operation

In practice a base-bias current $I_{B'E}$ is chosen that sets $r_{b'e}$, $c_{b'e}$ and c_f to unique constant values that are determined by (2), (3) and (5). The response of the network to low-level input signals is then performed by linear analysis. A practical case is that in which the transistor is driven from a voltage source V_{in} through a series resistance R_x. The collector current is

$$i_c(t) = \frac{\alpha_{fe} v_{in}}{R_x + r_{bb'} + r_{b'e}} \cdot \left\{ 1 - \exp\left[-\frac{t}{T_B'\left(\frac{R_x + r_{bb'}}{R_x + r_{bb'} + r_{b'e}} \right)} \right] \right\}. \quad (30)$$

Note that T_B' is modified by a resistance ratio which approaches one as $R_x \rightarrow \infty$ in the constant-current generator case. In large-signal operation this ratio modifies T_B' at each instant, while the ratio itself varies as a function of $i_{B'E}$ due to $r_{b'e}$.

IV. Measurements

A. $V_{B'E}$ vs $I_{B'E}$ Measurement

The bridge network and X–Y plotter shown in Fig. 13 are used to plot the $V_{B'E}$ vs $I_{B'E}$ internal-junction characteristic. This is accomplished by nulling out the voltage drop across $r_{bb'}$.

1) *T_B Measurement:* The T_B measurement suggested by Beaufoy and Sparkes provides the product of the input $r_{b'e}$ and $c_{b'e}$ elements of the transistor in the common-emitter mode of operation. The actual circuit that was used for making the T_B measurements is shown in Fig. 14. Small-signal T_B vs I_E plots for some 2N414 junction transistors are shown in Fig. 15.

2) *T_B' measurement:*

a) T_B' is equal in value to one time constant (0–63 per cent) of the collector-current response to a base constant-current step. The time constant is measured with i_C ranging from cutoff to just saturation with the desired value of R_L in the collector circuit. It is suggested that plots of T_B' vs R_L with V_{CC} as a parameter be used in large-signal design problems (see Fig. 16).

b) Obtaining T_B' from small signal data:[4] A T_B' measurement is made as described in a). T_B is then calculated from

$$T_B = T_B' - R_L C_c \alpha_{fe}, \quad (31)$$

where C_c and α_{fe} are evaluated at their midrange voltage $V_{CC}/2$ and current $V_{CC}/2R_L$ levels. T_B' can now be obtained for different values of R_L with no further time constant measurements by using the linear properties of the curves in Fig. 15 from which the average slopes

$$M = \frac{\Delta T_B}{\Delta I_E} \text{ are derived.} \quad (32)$$

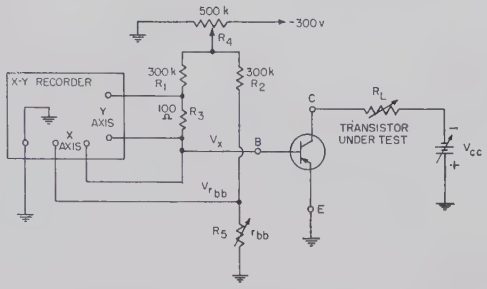


Fig. 13—$V_{B'E}$ vs $I_{B'E}$ measuring circuit.

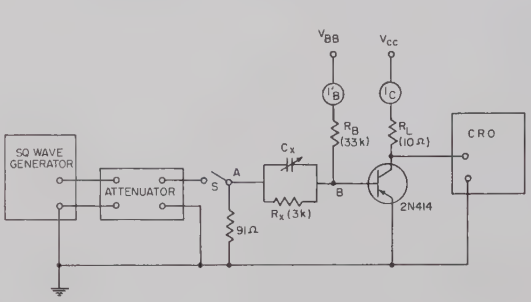


Fig. 14—Small-signal T_B measuring circuit for 2N414 junction transistor.

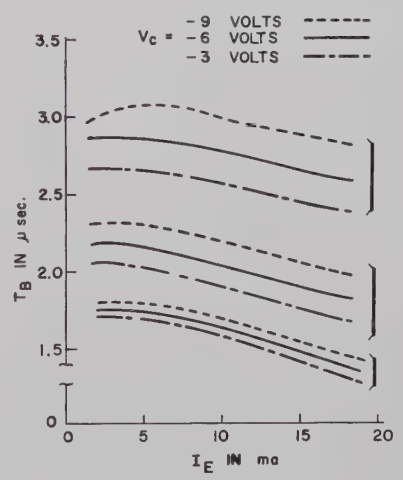


Fig. 15—Small-signal T_B vs I_E plots for 2N414 junction transistors.

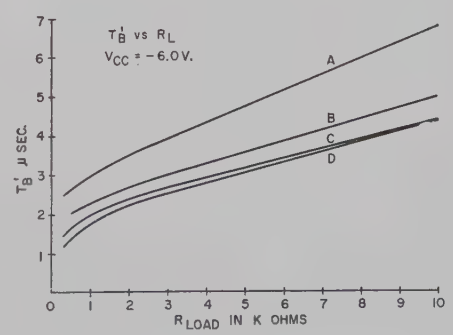


Fig. 16—T_B' vs R_L plots for 2N414 junction transistors.

When R_L is changed, the new T_B' is

$$T_B' = (T_B + M\Delta I_E) + R_L C_c \alpha_{fe}, \tag{33}$$

where ΔI_E is the difference in midrange emitter-current levels due to the new saturation level. T_B is the original reference value obtained from (31). C_c is unchanged and the new α_{fe} is obtained from curves such as shown in Fig. 17. Table II displays the experimental results of this method using nine 2N414 alloy junction transistors. The reference T_B was determined with $R_L = 300$ ohms. The calculated values of T_B' at $R_L = 1000$ ohms were then used to obtain the rise times in column 4.

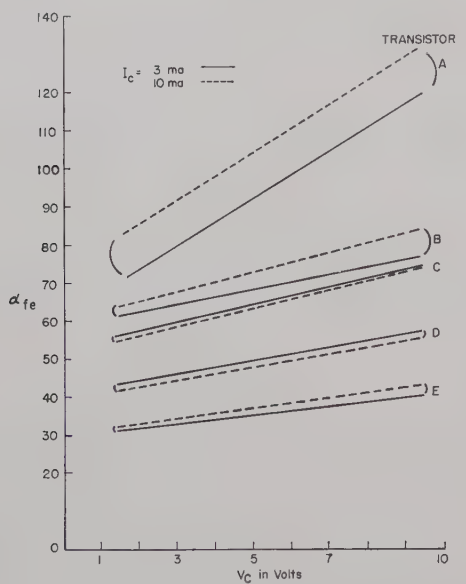


Fig. 17—α_{fe} vs v_c plots for 2N414 junction transistors.

TABLE II

Trans. 2N414	$M = \Delta T_B/\Delta I_E$	Measured Rise Time $R_L = 300$ ohms	Calculated Rise Time ($t_r = 2.2T_B'$) $R_L = 1000$ ohms	Measured Rise Time $R_L = 1000$ ohms
1	0.02 (10^{-3})	3.00 μsec	4.68 μsec	4.75 μsec
2	0.02 (10^{-3})	2.00 μsec	3.37 μsec	3.40 μsec
3	0.04 (10^{-3})	2.90 μsec	4.57 μsec	4.70 μsec
4	0.00 (10^{-3})	2.30 μsec	3.14 μsec	3.20 μsec
5	0.02 (10^{-3})	2.70 μsec	4.57 μsec	5.00 μsec
6	0.02 (10^{-3})	3.00 μsec	4.43 μsec	4.45 μsec
7	0.02 (10^{-3})	2.70 μsec	4.00 μsec	4.00 μsec
8	0.02 (10^{-3})	4.00 μsec	5.67 μsec	6.00 μsec
9	0.00 (10^{-3})	1.50 μsec	2.04 μsec	2.00 μsec

B. T_B' by Graphical Techniques

A graphical technique for determining T_B' is suggested by the linear decay of the base voltage $v_{B'E}$ that occurs when the transistor is current driven (Fig. 1). The base current during the decay cycle is

$$i_{B'E} = I_{B'E}(0)e^{-t/T_B'} \tag{34}$$

where $I_{B'E}(0)$ is the steady base current at the start of the decay cycle. Since $v_{B'E} = r_{B'E} i_{B'E}$, it can be shown from (4) and (38) that

$$v_{B'E} = V_{B'E}(0) - \frac{t}{39T_B'}. \tag{35}$$

Due to the linearity of the decaying $v_{B'E}$,

$$T_B' = -\frac{1}{39}\frac{\Delta t}{\Delta v_{B'E}}. \tag{36}$$

It is of interest to note that a similar result was produced from physical theory in a study of minority carrier lifetimes.[9]

V. Conclusion

Realistic analysis and simulation of large-signal transistor switching operation (nonsaturating) can be accomplished with four basic pieces of information: $V_{B'E}$ vs $I_{B'E}$, T_B', $r_{bb'}$ and α_{FE}. The use of the leakage current I_{CB0} is optional. The analysis is not restricted to the usual constant-current drive condition because the base-to-emitter nonlinear characteristics are included in the equivalent circuit. Figs. 1, 2, 9 and 10 show the variations in response as a function of different source impedances. For operations extending into saturation, two additional measurements K_s and r_s are required.

The general medium-speed equivalent circuit [Fig. 6(a)] provides the basis for more complex models. For example, the range of the circuit may be extended to higher speeds by adding the base-to-emitter space-charge capacitance in order to simulate the initial time delay effects.

The equivalent circuit is being employed at the present time for producing a transistor-circuit optimizing routine on an IBM-704 digital computer.

Appendix

The operation of numerous signal amplifying devices, such as the junction transistor, is described simply to a first order in terms of charge control theory.[10] This theory enables the time dependent characteristics of the device to be determined from its dc characteristics.

A basic relationship between the base control charge q_B and the associated base current $i_{B'E}$ for the medium-speed transistor is

$$\frac{q_B}{i_{B'E}} = T_B. \tag{37}$$

[9] S. R. Lederhandler and L. J. Giacoletto, "Measurement of minority carrier lifetime and surface effects in junction devices," Proc. IRE, vol. 43, pt. 1, pp. 477–482; April, 1955.

[10] E. O. Johnson and A. Rose, "Simple general analysis of amplifier devices with emitter, control and collector functions," Proc. IRE, vol. 47, pp. 407–418; March, 1959.

The B'–E junction V-I relationship can be expressed as[11]

$$i_{B'E} = \frac{I_{ES}}{(\alpha_{FE}+1)}\left[(\exp 39v_{B'E}) - 1\right], \tag{38}$$

where $\exp 39v_{B'E} \gg 1$ holds well in large-signal operation. The diffusion capacitance $c_{b'e}$ in Fig. 4, where

$$c_{b'e} = \frac{dq_B}{dv_{B'E}} = 39i_{B'E}T_B, \tag{39}$$

produces the dq_B/dt base charging current in (1).

The base current component dq_{BN}/dt which appears when the collector-to-base depletion-layer width varies as a function of v_C, is defined as

$$\frac{dq_{BN}}{dt} = C_c\frac{dv_C}{dt}. \tag{40}$$

This current can be associated with changes in $v_{B'E}$ rather than v_C by replacing C_c with a base-to-emitter capacitance c_f (Fig. 4); *i.e.*, since

$$dv_C = \alpha_{fe}R_L\frac{dq_B}{T_B}, \tag{41}$$

therefore,

$$c_f = \frac{dq_{BN}}{dv_{B'E}} = \left[\frac{R_LC_c\alpha_{fe}}{T_B}\right]\frac{dq_B}{dv_{B'E}}$$
$$= 39i_{B'E}R_LC_c\alpha_{fe}. \tag{42}$$

$r_{b'e}$ is found from (38) as

$$r_{b'e} = \left(\frac{di_{B'E}}{dv_{B'E}}\right)^{-1} = (39i_{B'E})^{-1}. \tag{43}$$

Eq. (43) agreed closely with the values obtained directly from the $V_{B'E}$ vs $I_{B'E}$ curve up to the experimental saturation base current level of $I_{B'ES}=0.15$ ma. For more exact $r_{b'e}$, $c_{b'e}$ and c_f values at high base-current densities, (38) may be rewritten as

$$i_{B'E} = \frac{I_{ES}}{(\alpha_{FE}+1)}\left[\left(\exp\left(\frac{39}{\lambda}\right)v_{B'E}\right) - 1\right], \tag{44}$$

where[12] $1<\lambda<2$ and $\lambda\to 2$ at high current densities. λ has been omitted since it is neither required for the analysis techniques developed in this paper, nor does it tend to affect the time constant T_B' significantly; *i.e.*,

$$r_{b'e}(c_{b'e}+c_f) = \frac{\lambda}{39i_{B'E}}\left(\frac{39i_{B'E}}{\lambda}T_B'\right) = T_B'. \tag{45}$$

Eq. (6) is derived from the nodal equations written on the network in Fig. 4, with $c_{b'e}$ and c_f defined by (39) and (42).

Explanation of Symbols

Upper case subscripts denote dc or total instantaneous values. Lower case subscripts denote instantaneous values with respect to a quiescent point. The terminal designations in Fig. 6(a) define the symbol subscripts.

α_{FES} = DC short-circuit current gain at saturation.
C_c = Collector-to-base space-charge capacitance. $C_c = A(V_{CB})^{-1/3}$ for alloy junction transistors.
$i_{B'E}$ = Resistive B'–E current.
$I_{B'ES}$ = Value of $i_{B'E}$ that produces saturation.
I_{CS} = Final collector current in saturation $(=V_{cc}/R_L+r_s)$.
I_{DS} = Excess base current in saturation that flows through D_1 $(I_{DS}=I_{B1}-I_{B'ES})$.
I_{ES} = Reverse emitter saturation current with the collector tied to the base.
q_B = Quantity of stored control charge in the base region (holes for a p-n-p transistor).
q_{BN} = Quantity of charge stored in the base region of a p-n-p transistor as impuirity electrons that are in equilibrium with the base donor ions.
T_B = Intrinsic time constant of the B'–E junction $(R_L=0)$.
T_B' = B'–E junction time constant which includes the effects of feedback capacitance. $T_B'=T_B+R_LC_c\alpha_{fe}$.
$R_{B'ES}$ = Final static value of the B'–E resistance R' at saturation.
r_s = Collector saturation resistance $= V_{CES}/I_{CS}$.
$V_{B'ES}$ = B'–E voltage at the onset of saturation. $V_{B'ES}$ is diode D_1 bias voltage.
V_{CES} = Final collector-to-emitter voltage in saturation.

Acknowledgment

The authors are grateful to R. D. Elbourn and J. A. Cunningham for their consultations and suggestions during the course of the study.

[11] W. Shockley, "The theory of p-n junctions in semiconductors and p-n junction transistors," *Bell Sys. Tech. J.*, vol. 28, pp. 435–489; July, 1949.
[12] R. N. Hall, "Power rectifiers and transistors," Proc. IRE, vol. 40, pp. 1512–1518; November, 1952.

Using Digital Computers in the Design and Maintenance of New Computers*

A. L. LEINER†, MEMBER, IRE, A. WEINBERGER†, MEMBER, IRE, C. COLEMAN†, AND H. LOBERMAN‡

Summary—**To conserve manpower and reduce the time needed for designing, assembling, and maintaining new large-scale computers, existing computers have been used to produce design and maintenance data for the new machines automatically. This paper describes some techniques that were devised at the National Bureau of Standards for automatically converting a logical design into detailed wiring plans which specify the actual pin-to-pin connections and other data necessary for assembling, debugging, and maintaining a new machine. Starting from punched-card input transcriptions of the logical design of the new computer, expressed in terms of Boolean algebra, an IBM 704 was programmed to produce 20,000 printed pages of wiring plans, fabrication data, and logical debugging and maintenance manuals.**

Introduction

In order to conserve scientific manpower and reduce the time needed for designing, assembling, and maintaining new large-scale digital computers, existing computers have been used in various ways to produce design data for the new machines automatically.[1–3] This paper describes some techniques that were devised at the National Bureau of Standards for mechanizing a sizeable portion of the design procedure for a new large-scale computer.[4,5]

The procedure for designing a large-scale computer system falls naturally into five general steps:

1) preparation of the system specifications for the new machine, indicating the operations that it is to perform and the speed that it is to attain,
2) organization of the system into large-scale functional blocks such as arithmetic units and control units,
3) development of the detailed logical design for each of these functional blocks,
4) conversion of this logical design into a detailed wiring plan from which the new computer can be fabricated, and
5) preparation of the maintenance manuals describing the logical and physical structure of the new computer in formats specially tailored to the needs of the debugging and maintenance staffs.

In deciding which steps of this procedure should be mechanized, the primary criterion employed was whether the machine program or the experienced computer designers could turn out a better end-product. In some cases it was decided that the machine program could *not* effectively compete with the empirical intuition of the experienced logical designer; in other cases, the machine permitted a degree of design sophistication, accuracy, and efficiency entirely unobtainable otherwise.

On the basis of this criterion, the fourth and fifth steps of the design procedure (namely, the conversion of the logical design into wiring plans, and the preparation of the maintenance manuals) were the main ones selected for mechanization. The third step (namely, the development of the detailed logical design) was carried out by the human designers with the aid of extensive computer-generated data. A brief summary of the automated procedure, carried out on an IBM 704, follows.

Starting from input transcriptions of the logical structure of the new system, expressed in terms of equations in Boolean algebra, machine programs subjected the input data to rigorous checking for internal consistency. For example, one type of consistency test verified that every signal used as an input to a given stage did in fact exist somewhere in the system as an output of another stage. Other programs checked the data for conformity with the numerous rules governing the desired usage of the electronic building blocks chosen to implement the system.

Next, using additional input data that described the physical locations in the equipment racks of the principal building blocks in the system, the 704 automatically assigned the much more numerous minor equipment components to optimum physical locations with respect to the principal ones. The machine then calculated the optimum wire routing between every connected set of components in the system, taking into account the various complicated restrictions on the lengths of wires that were imposed by the electronic

* Received by the PGEC, August 2, 1960. The work described here was carried out at the National Bureau of Standards, Washington, D. C.

† IBM Research Center, Yorktown Heights, N. Y.
‡ Data Processing Div., National Bureau of Standards, Washington, D. C.

[1] S. R. Cray and R. N. Kisch, "A progress report on computer applications in computer design," *Proc. Western Joint Computer Conf.*, San Francisco, Calif., February 7–9, 1956, AIEE, New York, N. Y.; 1956.

[2] M. Kloomok, P. W. Case, and H. H. Graff, "The recording, checking, and printing of logic diagrams," *Proc. Eastern Joint Computer Conf.*, Philadelphia, Pa., December 3–5, 1958, AIEE, New York, N. Y.; July, 1959.

[3] G. W. Altman, L. A. DeCampo, and C. R. Warburton, "Automation of computer panel wiring," *Communication and Electronics*, vol. 48, pp. 118–125; May, 1960.

[4] A. L. Leiner, *et al.*, "PILOT—a new multiple-computer system," *J. ACM*, vol. 6, pp. 313–335; July, 1959.

[5] A. L. Leiner, *et al.*, "PILOT—The NBS multicomputer system," *Proc. Eastern Joint Computer Conf.*, Philadelphia, Pa., December 3–5, 1958, AIEE, New York, N. Y.; July, 1959.

and mechanical properties of the components. After having determined the optimal wiring routing, the machine printed out a topological diagram of the minimum wiring tree, showing the physical location of each node and its connections with the other nodes in the tree. Finally, the machine calculated and printed out the exact lengths of all wires, the detailed pin-to-pin connections to be made, the type of cable to be used, the type of terminals to be attached, the text for labels on the wires, and detailed inventories of the components and materials used.

As an aid to debugging and maintaining the new computer, the 704 also generated detailed information on the interrelations between the logical structure of the system and the actual physical equipment used to implement the logic. This information includes data on the number of stages of each type driven by every stage, the total length of wire attached to each output, the length of the signal path from each output to its farthest destination, and an empirically calculated figure of merit measuring the degree of signal degeneration to be expected at each point of the wiring tree. These data can be used for comparing the observed operational performance of the finished computer with the predicted performance of its logical structure, circuitry, or components.

The input data for the IBM 704 consisted of 20,000 punched cards; the final output data consisted of the wiring plans and maintenance manuals for the new computer, which contains 7000 vacuum tubes, 165,000 solid-state diodes, and 70,000 pin-to-pin wiring connections. This machine-generated output data comprises approximately 20,000 printed pages. The total machine time used for testing and running the program was approximately 175 hours.

The over-all automated procedure is illustrated in the block diagram of Fig. 1. Figs. 2, 5, 7, 9, and 10 illustrate the blocks of Fig. 1 in greater detail.

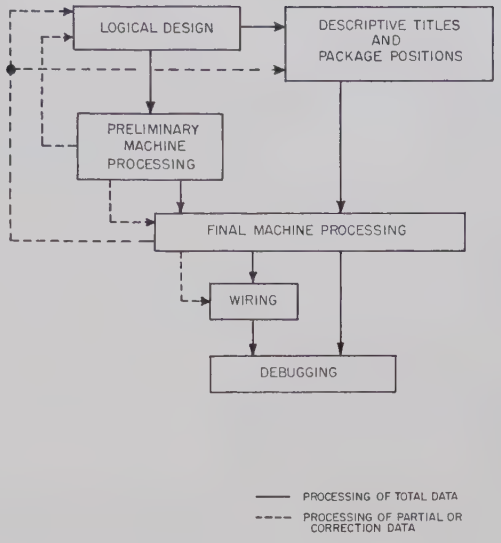


Fig. 1—Over-all flow of automated procedure.

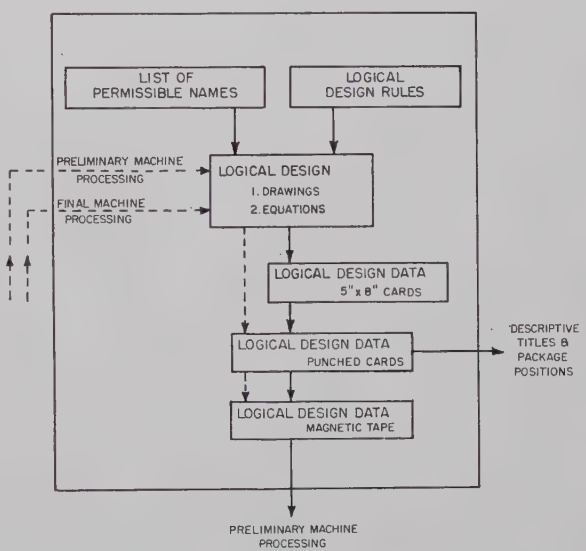


Fig. 2—Logical design procedure.

Logical Design Procedures

In formulating the logical design procedures illustrated in Fig. 2, an effort was exerted to make the new automated techniques conform to the existing habits of the logical designers rather than vice versa. One problem that arose in trying to couple the human designers to an automated procedure was that of notation. In assigning names to the signals transmitted between the various parts of computers, designers generally like to invent picturesque mnemonic words to help them in remembering the functions of the various signals. Automatic techniques, on the other hand, demand some sort of standardized nomenclature to keep things from getting out of hand.

These conflicting needs were reconciled by devising a technique for generating pronounceable names on a computer by means of the formula illustrated in Fig. 3. Lists of these pronounceable names were generated automatically, were printed out, and were given to the designers in advance of their work on the logical design of the new machine. From these lists, the designers could then choose suitable names for signals; see the sample list in Table I. Because the signal names were pronounceable, the designers found them easy to remember and to talk about in the course of their design work.

The standard names also protected the designers against various kinds of clerical mistakes, keypunching errors, and machine errors in the following way. Of all the possible combinations of characters from which the names could be formed, the formula selected only a small percentage. Therefore, by putting all the input data through a machine check which tested for conformity to the formula, the various kinds of random errors in the signal names could be screened out, with

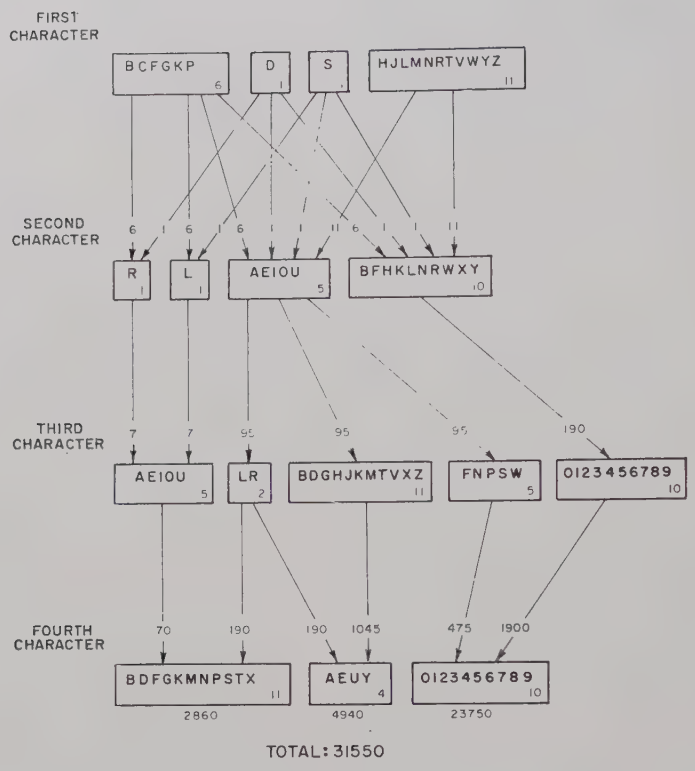


Fig. 3—Formula for generating signal names.

TABLE I

SAMPLE LIST OF SIGNAL NAMES

BABA	BABE	BABU	BABY	BADA	BADE	BADU	BADY	BAGA	BAGE
BAGU	BAGY	BAHA	BAHE	BAHU	BAHY	BAJA	BAJE	BAJU	BAJY
BAKA	BAKE	BAKU	BAKY	BALA	BALB	BALD	BALE	BALF	BALG
BALK	BALM	BALN	BALP	BALS	BALT	BALU	BALX	BALY	BAMA
BAME	BAMU	BAMY	BARA	BARB	BARD	BARE	BARF	BARG	BARK
BARM	BARN	BARP	BARS	BART	BARU	BARX	BARY	BATA	BATE
BATU	BATY	BAVA	BAVE	BAVU	BAVY	BAXA	BAXE	BAXU	BAXY
BAZA	BAZE	BAZU	BAZY	BEBA	BEBE	BEBU	BEBY	BEDA	BEDE
BEDU	BEDY	BEGA	BEGE	BEGU	BEGY	BEHA	BEHE	BEHU	BEHY
BEJA	BEJE	BEJU	BEJY	BEKA	BEKE	BEKU	BEKY	BELA	BELB
BELD	BELE	BELF	BELG	BELK	BELM	BELN	BELP	BELS	BELT
BELU	BELX	BELY	BEMA	BEME	BEMU	BEMY	BERA	BERB	BERD
BERE	BERF	BERG	BERK	BERM	BERN	BERP	BERS	BERT	BERU
BERX	BERY	BETA	BETE	BETU	BETY	BEVA	BEVE	BEVU	BEVY
BEXA	BEXE	BEXU	BEXY	BEZA	BEZE	BEZU	BEZY	BIBA	BIBE
BIBU	BIBY	BIDA	BIDE	BIDU	BIDY	BIGA	BIGE	BIGU	BIGY
BIHA	BIHE	BIHU	BIHY	BIJA	BIJE	BIJU	BIJY	BIKA	BIKE
BIKU	BIKY	BILA	BILB	BILD	BILE	BILF	BILG	BILK	BILM
BILN	BILP	BILS	BILT	BILU	BILX	BILY	BIMA	BIME	BIMU
BIMY	BIRA	BIRB	BIRD	BIRE	BIRF	BIRG	BIRK	BIRM	BIRN
BIRP	BIRS	BIRT	BIRU	BIRX	BIRY	BITA	BITE	BITU	BITY
BIVA	BIVE	BIVU	BIVY	BIXA	BIXE	BIXU	BIXY	BIZA	BIZE
BIZU	BIZY	BLAB	BLAD	BLAF	BLAG	BLAK	BLAM	BLAN	BLAP

odds ranging from 20:1 to 500:1. In some common cases, such as the interchange of the letter "O" and the numeral "0," etc., the program was instructed to make the proper corrections automatically.[6]

The logical design was formulated by the designers using either gating diagrams or Boolean equations, according to their individual habits or the intricacy of the part of the system being worked on. These diverse notations were then brought together into a standard format by clerks who transcribed the data from the designers' original plans on to standard forms printed on 5″×8″ file cards. See Fig. 4 for an illustration of such a design data card, which contains the name of a tube-amplifier signal stage (RIF2), the drawing number from which the data was copied (R-140), and the names of the input signals to the stage. For example, the Boolean equation here represented is

$$\begin{aligned} \text{RIF2} = {} & (\overline{\text{REXE}})(\text{RH79})(\text{RW07})(\overline{\text{RW08}}) \\ & + (\overline{\text{REXE}})(\text{RIHA})(\text{RON5})(\text{REF1})(\overline{\text{REF2}}) \\ & + (\overline{\text{REXE}})(\text{RH80})(\text{REF1})(\overline{\text{REF2}}) \\ & + (\text{ROF5})(\overline{\text{RW08}})(\overline{\text{RW13}})(\overline{\text{RW14}})(\overline{\text{RW15}})(\text{RW16}). \end{aligned}$$

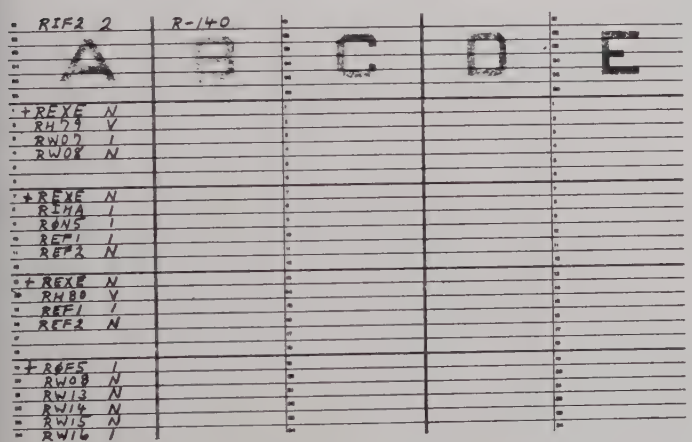


Fig. 4—Example of a data design card.

Concerning the symbols following the names of the input signals on the design card, the letter N means that the negative or dual output of the tube-amplifier stage is used, the letter V indicates that the signal is an OR-gate output, and the numeral 1 indicates the positive output of a tube-amplifier stage.

For checking purposes, the clerical transcribers maintained tallies on the data entered, such as: the number of AND-gate inputs, OR-gate inputs, positive signals, negative signals, etc. This tally data was subject to independent recalculation by the machine program later on, and all discrepancies reconciled before proceeding.

The logical design cards then served as copy for the keypunchers, who transcribed all the indicated information verbatim. Transcribed to magnetic tape, the data constituted the main design file describing the logic of the system.

Generation of Wiring Plans

The logic associated with a typical package is shown in Fig. 6, illustrating a designer's original working drawing. The four AND-gates and the tube-transformer stage labeled RIF2 driven by their mixed outputs constitute a single package. The OR-gates labeled RH79 and RH80 respectively, are contained in separate packages, each holding twelve independent two-input OR-gates.

Tube-transformer stage packages (7000 in all) were positioned manually by the logical designers, who also specified the positions of the packages containing OR-gates. On the other hand, the assignment of 5000 individual OR-gates to positions within the available OR-gate package locations was carried out by the machine program, which allocated each OR-gate as nearly as possible to an unassigned position midway between the locations of its input signal sources and its output destinations. The assignment of other minor components was also carried out automatically. This distribution of responsibility between the designers and the computer program was chosen largely in order to simplify maintenance procedures. It was felt that the designers' broad over-all knowledge of the logical interrelationships between the various parts of the system would lead to more natural placement patterns than a strictly automatized process.

Some of the principal processing tasks directed towards producing detailed wiring data are listed in Fig. 7. The lengths of all the possible pin-to-pin wiring routes (branches) between the points to be connected were calculated, and from among all the possible routings those branches were selected which minimized the total wire length (subject to various rules and constraints established for the different types of signals being transmitted).[7] The various branches in the selected wiring tree were then printed out as pin-to-pin wiring tables which specify the positions of both ends of each branch, the name of the signal being transmitted, the name of the destination, the length of the wire, and the types of terminals to be used (see Fig. 8 and, for details, Table II). These lists also provided copy for labels to be affixed to the branches. Additional processing tasks included the production of summary statistics, such as the required number of wires in each length category, etc.

[6] A. L. Leiner and W. W. Youden, "A system for generating pronounceable names using a computer," *J. ACM*, vol. 8, pp. 97–103; January, 1961.

[7] H. Loberman and A. Weinberger, "Formal procedures for connecting terminals with a minimum total wire length," *J. ACM*, vol. 4. pp. 428–437; October, 1957.

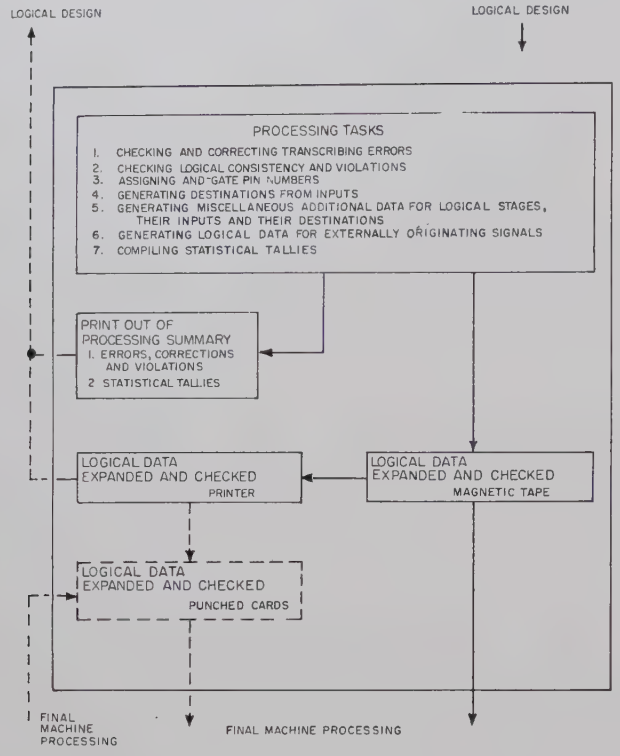
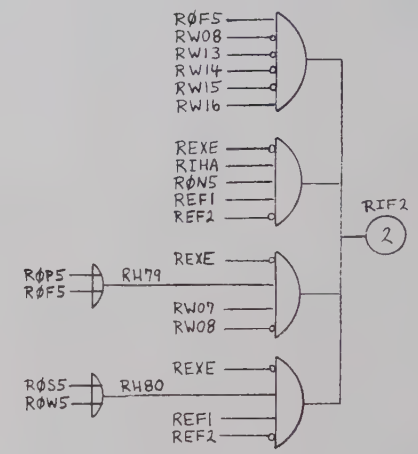


Fig. 5—Preliminary machine processing.



$$RIF2 = R\emptyset F5 \cdot \overline{RW08} \cdot \overline{RW13} \cdot \overline{RW14} \cdot \overline{RW15} \cdot RW16 + \overline{REXE} \cdot RIHA \cdot R\emptyset N5 \cdot REF1 \cdot \overline{REF2} + \overline{REXE} \cdot RH79 \cdot RW07 \cdot \overline{RW08} + \overline{REXE} \cdot RH80 \cdot REF1 \cdot \overline{REF2}$$

$$RH79 = R\emptyset P5 + R\emptyset F5$$

$$RH80 = R\emptyset S5 + R\emptyset W5$$

Fig. 6—Example of a designer's working drawing.

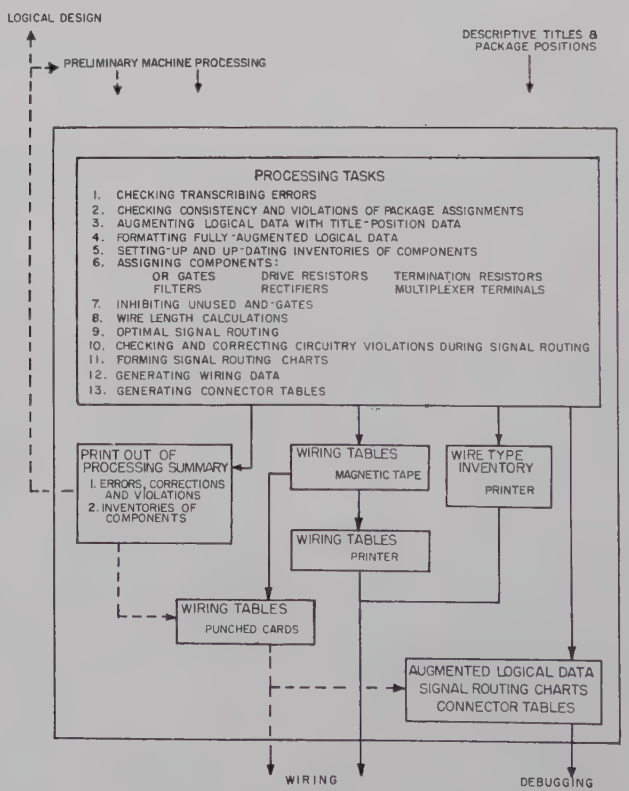


Fig. 7—Final machine processing.

```
014000 C3W39 R1F22 C9S24 RL73V 043.0 TT   39350
011000 C9S24 R1F22 C9S06 RL75V 004.3 TT   39351
024000 C3W43 R1F22 D7C09 RUN24 037.0 TT   39352
021000 D7C09 R1F22 D7C13 RUN24 003.0 TT   39353
021000 D7C13 R1F22 D8F13 RAP24 012.5 TT   39354
023000 D8F13 R1F22 C9Q14 RELA3 032.0 TT   39355
024000 C3W43 R1F22 D3Q01 NNO54 037.0 TT   39356
021000 D3Q01 R1F22 D3R01 NNO64 003.5 TT   39357
021000 D3R01 R1F22 D3S01 NNQ74 003.5 TT   39358
021000 D3S01 R1F22 D3T01 NNO84 003.5 TT   39359
021000 D3T01 R1F22 D3V01 NN214 005.0 TT   39360
021000 D3V01 R1F22 D3W01 NN224 003.5 TT   39361
021000 D3W01 R1F22 D3X01 NN234 003.5 TT   39362
021000 D3X01 R1F22 D3Y01 NN244 003.5 TT   39363
023000 IAU10 R1F22 I3Y18 JON24 032.0 ET 1 39364
023000 D3Q01 R1F22 DAQ03 MPXR* 027.0 TE 2 39365
024000 DAQ03 R1F22 IAU10 MPXR* 072.0 EE 3 39366
021000 C3W43 R1F22 C3W41 RES.* 003.0 TT 7 39367
021000 C3W41 R1F22 C3W40 RES.* 003.0 TT 7 39368
021000 I3Y18 R1F22 I3Y40 TRMN* 004.6 TT 8 39369
031000 C3W29 R1F22 C4T02 RUF23 011.0 TT   39370
031000 C4T02 R1F22 C4T08 RUF23 003.0 TT   39371
031000 C3W29 R1F22 C2S15 RUS23 011.8 TT   39372
032000 C2S15 R1F22 C1Q02 RUW23 015.5 TT   39373
033000 C4T08 R1F22 C7X20 RIS23 029.0 TT   39374
033000 C4T08 R1F22 C5C02 RIW23 030.0 TT   39375
031000 C5C02 R1F22 C5C10 RIW23 003.0 TT   39376
034000 C5C10 R1F22 C1V32 TRMN* 052.0 TT 8 39377
```

Fig. 8—Example of a wiring table (see Table II).

TABLE II

GUIDE TO WIRING TABLE (See Fig. 8)

Column	Meaning
1	Code for signal type and wire length class
2	Position (rack, row, column) of end of wire branch nearest signal source
3	Name of stage (source of signal)
4	Position (rack, row, column) of end of wire branch nearest signal destination
5	Name of destination stage
6	Length of wire branch in inches
7	Type of terminals (taper-pin or edge-on)
8	Memory code
9	Serial number of branch

GENERATION OF MAINTENANCE MANUALS

As an aid to understanding the over-all logic of the system, the machine was subdivided into ten major blocks, which were further broken down into 300 units and 1700 subunits. Each subunit was given a descriptive title containing up to 45 alphanumeric characters, and a 6-digit code number. The assignment of stages to subunits was made by the designers in the course of laying out the logic. Later on, at the time the stages were assigned to physical locations in the racks, the subunit titles were also affixed to the stage record (see Fig. 9). These descriptive titles were carried with the stage record throughout the subsequent processing and appear in the final maintenance records.

Three principal types of final maintenance records are illustrated. In Fig. 11 is shown a sample of an input-destination table. Table III gives a detailed explanation of the format. These tables show, for every stage: the stage name, drawing number, physical position, descriptive subunit title; the input signal logic with names, positions and drawing numbers of the sources of all incoming signals; the load on each winding of the output transformer of the stage, the names of the stages that it drives, and their positions and drawing numbers.

Associated with each such input-destination table is a tree diagram table, illustrated in Fig. 12. The tree diagram illustrates the topology of the wiring tree which distributes the output signals of the stage. Each line of print on the diagram contains the pin position of one end of a wire branch, the name of the signal originating in that package, the length of that branch, the cumulative wire distance from the stage output to that point, and the "load-length product" at that point. The "load-length product" is a figure of demerit proportional to the sum of the products of the currents and branch lengths over the path from the stage output pin to that point. A higher than average value for this parameter suggests that a more than average pulse-shape degeneration should be expected. Table IV gives a detailed explanation of Fig. 12.

A third type of record is the connector table, illustrated in Figs. 13–15 (see also Tables V–VII). The format of these tables is isomorphic with the actual appearance of the 44-pin package receptacle (connector) as seen from the back-wiring side of the racks. Besides the identifying position and title data already described, the table indicates for each of the 44 pins the name of the incoming signal and the position of the other end of the branch. If the signal proceeds out to another pin, the record also shows the name of the stage located at the other end of the branch and its pin position. Power wiring with color codes also is shown. For OR-gate packages, a table is given at the top of the record identifying by name the OR-gate located in each segment of the package (see Fig. 15 and Table VII for details).

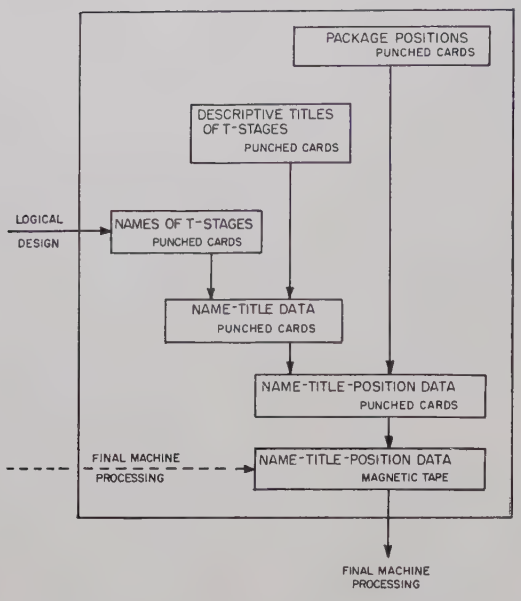


Fig. 9—Descriptive titles and package positions.

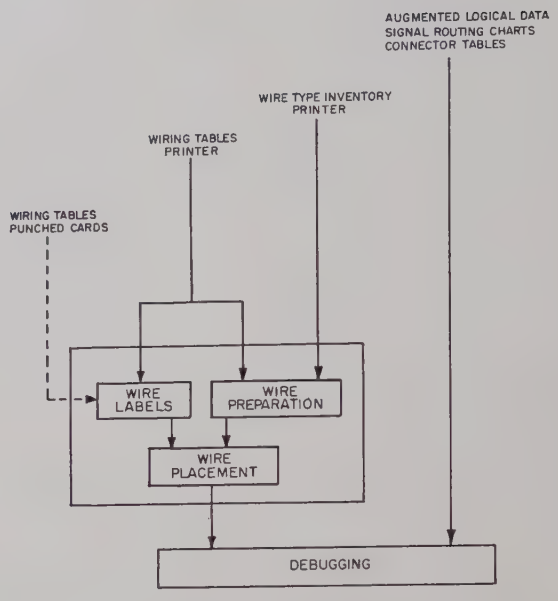


Fig. 10—Wiring data.

```
RIF22(140)C3W(S1 ) 2330601SCU MEM CAB 2 SEL. G OR NEXT SEQ INSTR SEL.   RIF22

   (6)AND-GATE1  +006(007)NEGATIVE +013(013)INDIRECT +002(002)DIRECT=21
ROF55(137)C4L*01  RIS23(140)C7X20   JON24(147)I3Y18  RL73V1,C9S24(003.0,003.0,0)
RW08N(133)D6H*02  RIW23(140)C5C02   NN054(101)D3Q01        =RELB3(148) C9R  3
RW13N(133)D5J*03  RIW23(140)C5C10   NN064(101)D3R01  RL75V1,C9S06(003.0,003.0,0)
RW14N(133)D5K*04  RUF23(149)C4T02   NN074(101)D3S01        =RELB3(148) C9R  3
RW15N(133)D5L*05  RUF23(149)C4T08   NN084(101)D3T01
RW16I(133)D5M*06  RUS23(149)C2S15   NN214(101)D3V01
   (5)AND-GATE2   RUW23(149)C1002   NN224(101)D3W01
REXEN(138)C3J*07                    NN234(101)D3X01
RIHA5(154)C9V*08                    NN244(101)D3Y01
RON55(137)C90*09                    RAP24(160)D8F13
REF11(143)C4M*10                    RELA3(148)C9Q14
REF2N(145)C4M*11                    RUN24(136)D7C09
   (4)AND-GATE3                     RUN24(136)D7C13
REXEN(138)C3J*12
RH79V      000*13
RW071(133)D6G*14
RW08N(133)D6H*15
   (4)AND-GATE4
REXEN(138)C3J*17
RH80V      000*18
REF11(143)C4M*19
REF2N(145)C4M*20
```

Fig. 11—Example of input-destination table (see Table III).

TABLE III

GUIDE TO INPUT-DESTINATION TABLE (SEE FIG. 11)

Line	Entry	Meaning
	RIF22(140)	Stage name, clock phase, drawing no. "(140)"
	C3W	Position (Rack, row, column)
1	(S1) → (a) (b) (c)	a. Type of Stage: S = Single-tube D = Double-tube b. Phase-skipped inputs 1 = Has a phase-skipped input Blank = No phase-skipped input c. Rectifier R = Has rectifier Blank = No rectifier
	2330601SCU MEM CAB 2 SEL. → (a) (b) (c)	a. Memory code b. Unit-subunit No. c. Unit title
	G OR NEXT SEQ INSTR SEL.	Subunit title

TABLE III, *cont.*

Line	Entry	Meaning
	(6) AND-GATE 1	Six "(6)" Inputs into AND-gate No. 1
2	+006(007)NEGATIVE (a) (b)	a. No. of loads on negative output signal b. No. of destinations of negative output signal
	+013(013)INDIRECT (a) (b)	a. Drive of indirect output signal b. No. of destinations of indirect output signal
	+002(002)DIRECT=21 (a) (b) (c)	a. No. of loads on direct output signal b. No. of OR-gate destinations on direct output signal c. Total load on stage
3	ROF55(137)C4L*01 (a) (b) (c)	a. Name, clock phase and drawing no. of input to AND-gate b. Position of input stage "ROF5" c. Pin no. on AND-gate of "RIF2"
	RIS23(140)C7X20 (a) (b)	a. Name, clock phase, and drawing no. of destination of negative signal b. Position and pin no. at destination
	JON24(147)I3Y18 (a) (b)	a. Name, clock phase and drawing no. of indirect signal destination b. Position and pin no. at destination
	RL73V1, C9S24(003.0, 003.0, 0) (a) (b) (c) (d) (e) (f)	a. Name of OR-gate which is a direct signal destination b. Drive of OR-gate c. Input pin no. on "RL73" from direct signal of "RIF2" d. Longest line length (inches) of OR-gate tree of "RL73" e. Total branch length (inches) of OR-gate tree of "RL73" f. Branching number of the output-terminal of the OR-gate "RL73"
4	RW08N (a) (b)	a. Name of second input to first AND-gate b. Negative signal clock phase N = Neg. clock phase 1 O = Neg. clock phase 2 P = Neg. clock phase 3 Q = Neg. clock phase 4 R = Neg. clock phase 5
	RIW23(140)C5C02	Name, clock phase, drawing no., position, and pin no. of second negative destination
	=RELB3(148) C9R 3 (a) (b) (c) (d)	a. Shows beginning of OR-gate destination b. Name, clock phase, and drawing no. of destination of OR-gate "RL73" (Final destination of tube stage "RIF2") c. Position (Rack, row, column) of "RELB" d. Type of stage ("RELB") 1 = D (upper half of double-tube) 2 = E (lower half of double-tube) 3 = S (single tube)
9	(5)AND-GATE2	Five "(5)" inputs to AND-gate No. 2
17	RH79V 000*13 (a) (b) (c)	a. Name of OR-gate input b. No meaning c. Pin no. on AND-gate

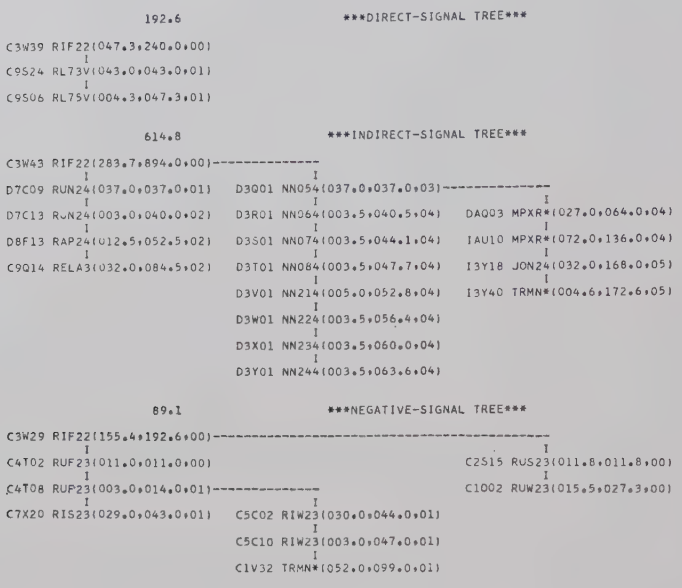


Fig. 12—Example of a tree-diagram table (see Table IV).

TABLE IV

GUIDE TO TREE DIAGRAM TABLE (See Fig. 12)

Line	Entry	Meaning
1	192.6	Additional allowable branch length (inches) for the direct-signal tree of "RIF2." This is the difference between the maximum allowable total tree branch length and the actual total tree branch length.
2	C3W39 RIF22 (a) (b)	a. Position and output-terminal no. b. Name and clock phase of the tube-stage which is the source of the direct-signal tree.
	(047.3, 240.0, 00) (a) (b) (c)	a. Actual total tree branch length in inches. b. Maximum allowable total tree branch length in inches. c. Load-length product for this terminal in units of 100 load-inches.
3	I	Used to show routing of branches between terminals (pins).
4	C9S24 RL73V	Position, pin no., and OR-gate name of direct-signal destination
	(043.0, 043.0, 01) (a) (b) (c)	a. Branch length (inches) between output-terminal (C3W39) and this terminal (C9S24) b. Line length (inches) from output-terminal (C3W39) to this terminal (C9S24) c. Load-length product for this terminal
6	C9S06 RL75V	Position, pin no., and OR-gate name of direct-signal destination
	(004.3, 047.3, 01) (a) (b) (c)	a. Branch length between terminals C9S24 and C9S06 b. Line length between output-terminal C3W39 and C9S06 c. Load-length product for this terminal
7	614.8	Additional allowable branch length (inches) for the indirect-signal tree of "RIF2"
8	— — — — —	Used to show routing of branches between terminals (pins)
12	DAQ03 MPXR*	Position and pin no. of multiplexer ("MPXR)" terminal
18	I3Y40 TRMN*	Position and pin no. of termination ("TRMN") terminal

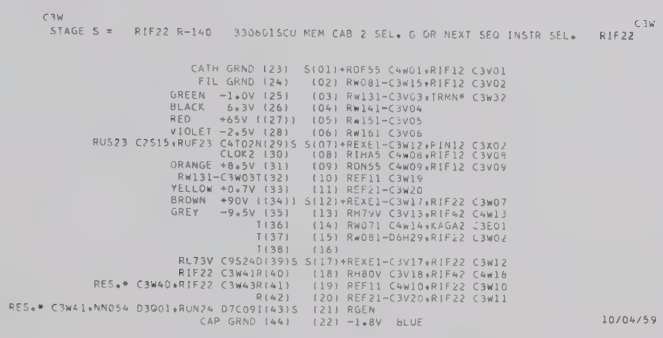


Fig. 13—Example of a connector table, single-tube package (see Table V).

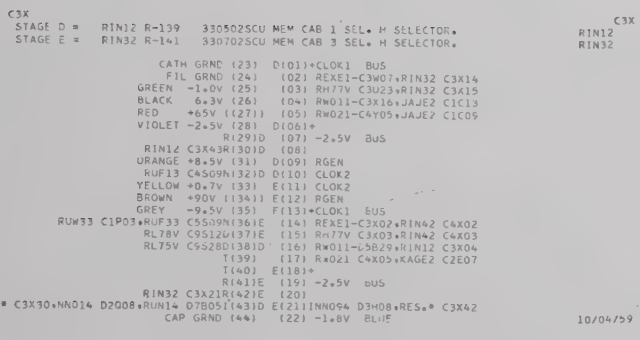


Fig. 14—Example of a connector table, double-tube package (see Table VI).

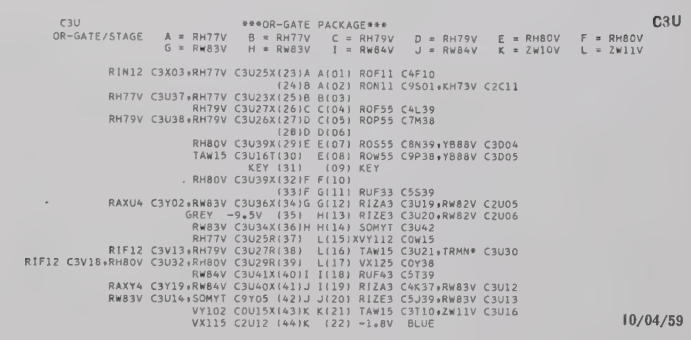


Fig. 15—Example of a connector table, OR-gate package (see Table VII).

TABLE V

GUIDE TO CONNECTOR TABLE (SINGLE-TUBE PACKAGE) (See Fig. 13)

Line	Entry	Meaning
1	C3W	"C" = Rack C "3" = Row 3 "W" = Column W
2	RIF22 R-140	Name, clock phase, drawing no. of stage
	330601SCU MEM CAB 2 SEL. G OR NEXT SEQ INSTR SEL.	Unit-subunit no. and title
3	CATH GRND (23)	Cathode ground on pin no. 23
	S	Single-tube
	+	Indicates first pin of AND-gate
	ROF55 C4W01	Name, clock phase, position, and pin no. of input signal branch
4	FIL GRND (24)	Filament ground on pin no. 24
5	GREEN −1.0V (25)	Green colored −1.0 volt power lead on pin no. 25
	RW131-C3VO3	"−" indicates negative signal
	,TRMN* C3W32	Branch goes from pin no. 03 to termination at position C3W32
7	RED+65V ((27))	Double parentheses indicate high voltage present
9	,RUF23 C4T02N(29)	*Negative* ("N") output branch goes to "RUF2" at C4T02 from pin no. 29
10	CL0K2 (30)	Normal clock phase 2 on pin no. 30
12	RW131-C3W03T(32)	Pin no. 32 provides a negative termination ("T") for stage "RW13" from C3W03
16	T(36)	Pin no. 36 provides a positive termination ("T")

TABLE IV, *cont.*

Line	Entry	Meaning
19	RL73V C9S24D(39)	*Direct* ("D") output branch goes to OR-gate "RL73" at C9S24
20	RIF22 C3W41R(40)	Pin no. 40 is connected to an internal extra drive resistor ("R") which is connected to "RIF2" at C3W41
23	RES.* C3W41	Indirect output is connected to extra drive resistor (RES.*) at C3W41
	,NN054 D3Q01, RUN24 D7C09I(43)	From *Indirect* output ("I") pin no. 43 are branches to "NN05" at D3Q01 and "RUN2" at D7C09
	(21) RGEN	Pin no. 21 is for regeneration
24	CAP GRND (44)	Capacitance ground on pin no. 44
	10/04/59	Date of connector table run

TABLE VI

GUIDE TO CONNECTOR TABLE (DOUBLE-TUBE PACKAGE) (See Fig. 14)

Line	Entry	Meaning
2	STAGE D=	"D" indicates upper half of double-tube package
3	STAGE E=	"E" indicates lower half of double-tube package
4	D(01)+CLOK 1 BUS (a) (b)	a. Pin no. 1 is the first pin of the first AND-gate for stage "D" (RIN1). All pins labeled "D" or "E" apply to stages "D" or "E" respectively b. Added clock phase 1 from bus connected to pin no. 1
10	(07) −2.5 BUS	−2.5 Volt inhibit branch from power bus to pin no. 7
24	E(21)I NN094 D3H08, RES.*C3X42	Stage "E" (RIN3) has an Indirect output ("I") to stage "NN09" at D3H08 and also is connected to an extra drive resistor (RES.*) at C3X42

TABLE VII

GUIDE TO CONNECTOR TABLE (OR-GATE PACKAGE) (See Fig. 15)

Line	Entry	Meaning
2	A=RH77V · · · F=RH80V	Pins labeled "A" on connector apply to OR-gate stage "RH77." "V" indicates OR-gate. Pins labeled "F" apply to OR-gate stage "RH80," etc.
4	RIN12 C3X03, RH77V C3U25X(23)A	For OR-gate stage "A" (RH77) pin no. 23 is the output ("X") and there is a branch to tube stage "RIN1" at C3X03. The branch to OR-gate "RH77" at C3U25 combines two physical OR-gates for added drive
12	KEY(31) (09) KEY	This connector has a key at pin nos. 31 and 09 to permit the insertion of only an OR-gate package
18	RH77V C3U25R(37)	Pin no. 37 provides an extra drive resistor ("R") for OR-gate "RH77 at C3U25

CONCLUSION

In designing new large-scale machines, computer designers have become increasingly burdened with the sheer volume of routine and tedious tasks involved in preparing accurate wiring data, assembly instructions and maintenance guides from logical design data. To alleviate this burden it was only natural for computer designers to turn to their own handiwork, the computer itself, for aid. It was in fact poetic justice to demonstrate in our own work the highly touted claim of computers to relieve tedium.

In carrying out these tedious tasks, the computer was used as a tool for aiding the human engineers rather than as an instrument for replacing them. The over-all effort was viewed as one of mutually adapting the human engineers and the automatic processing equipment in order to take full advantage of those well-developed human skills and intuitions that are not yet well enough understood to be formulated in terms of programmable operations.

ACKNOWLEDGMENT

The authors wish to acknowledge the valuable contributions to this work by W. Hall and J. Beiman, who wrote codes for the preliminary machine processing, and by W. W. Youden, who wrote the code for generating the signal names.

Skip Techniques for High-Speed Carry-Propagation in Binary Arithmetic Units*

M. LEHMAN† AND N. BURLA‡

Summary—After a very brief summary of the various well-known methods of expediting carry-propagation in binary arithmetic units, the paper discusses and develops the "anticipated-carry" or "carry-skip" technique originally due in decimal form to Babbage, much used in mechanical calculators and lately revived for use in binary units. Various degrees of refinement are possible. It appears that for a given expenditure, the technique results in a unit which is simpler and faster than those using one of the other techniques. Conversely in order to attain a given speed with given circuit elements, the skip technique appears to minimize the equipment requirement among the known speed-up techniques.

I. Introduction

THE central problem in the design of high-speed arithmetic units is the minimization of the time which has to be allowed for the possible propagation of carry in the adder network. Various solutions to this problem have been proposed. In the asynchronous *carry completion detection* system due to Gilchrist, Pomerene and Wong,[1] the actual completion of the propagation process is detected. A signal derived from the detection circuit may then be used to initiate the remainder of the addition process. All waiting time and margins of safety in the control circuits may therefore be eliminated. Adders based on this principle are efficient since it has been shown[2] that the average length of the maximum propagation chain in an n-bit binary adder is less than $\log_2 n$.

Weinberger and Smith in their *simultaneous carry* circuit[3,4] take advantage of the fact that the recursive form of the full carry-function may (for an adder not using the *ones complements* representation for negative numbers) be expressed in nonrecursive (or partially-recursive) form. This implies that a circuit realization of the carry-function may, in each stage, determine the carry as an explicit function of all (or part of) the less significant operand bits, thus eliminating (or reducing) the carry-propagation.

A solution based as much on circuit techniques as on logical considerations is due to Kilburn, Edwards and Aspinall.[5] By expressing the carry-function in disjunctive form, with mutually exclusive conjunctive terms, it becomes possible to replace the OR gate of the normal circuit with a simple junction of conductors, thus halving the number of gates through which a propagated carry-signal has to pass. Furthermore, by selecting working conditions so that the transistor switches operate between two current levels rather than in the ON-OFF mode, the *exclusive OR* circuit achieves minimization of carry time.

A further method of speeding up mechanized addition appears to be less well-known. This is the *pyramid circuit* described by Nadler.[6] Addend and augend bits are added to produce partial (half) sum and carry bits in a half adder, and are temporarily stored. The carry bits are then assimilated in a series of steps, during each of which only every second carry position, still existing, is absorbed. Each carry thus absorbed is allowed to *propagate* as far as the next stage at which a carry may still exist. This propagation does not however result in a propagation delay. The path consists of a series of AND gates controlled only by the partial sum bits, the assimilation circuits being analogous to very simple simultaneous (see above) or skip (see below) carry circuits. The number of places in which no carry can exist is doubled in each phase and the final sum is determined in a time proportional to the logarithm of the number length.

Sklansky's *conditional sum adder* recently described[7] is, at first sight, similar to Nadler's[6] circuit. Closer examination reveals that the fundamental philosophy is however quite different and appears rather more complex. It is based on the determination of the sums conditional and output-carries that can arise from all possible distributions of input-carries for groups of addend and augend bits. By passing the appropriate in-

* Received by the PGEC, December 8, 1960; revised manuscript received, May 18, 1961. The circuits and techniques discussed here were to the authors' best knowledge first published by N. Burla in his Master's Thesis,[14] June, 1960. Since this paper was submitted, O. L. MacSorley[15] has described similar techniques.

† Science Dept., Israel Ministry of Defence, Hakirya, Tel Aviv, Israel.

‡ Israel Atomic Energy Commission, Tel Aviv, Israel. Formerly The Technion, Israel Institute of Technology, Haifa, Israel.

[1] B. Gilchrist, J. J. Pomerene, and S. Y. Wong, "Fast carry logic for digital computers," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-4, pp. 133–136; December, 1955.

[2] A. W. Burkes, H. H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," pt. 1, Institute of Advanced Study, Princeton University, N. J.; 1947.

[3] A. Weinberger and J. L. Smith, "A one microsecond adder using one megacycle circuitry," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-5, pp. 67–73; June, 1956.

[4] A. Weinberger and J. L. Smith, "A Logic for High Speed Addition," Natl. Bur. Standards, Washington, D. C., Circular 591, Sec. 1; February, 1958.

[5] T. Kilburn, D. B. G. Edwards, and D. Aspinall, "Parallel addition in digital computers, a new fast carry circuit," *Proc. IEE*, vol. 106, pt. B, pp. 464–466; September, 1959.

[6] M. Nadler, "A high speed electronic arithmetic unit for automatic computing machines," *Acta Tech.* (of the Czechoslovak Academy of Science?), no. 6, pp. 464–478; 1956.

[7] J. Sklansky, "Conditional-sum addition logic," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-9, pp. 226–231; June, 1960.

formation through a series of networks or selecting switches, the number of such bits determined unconditionally may be doubled in each of a series of steps. Here also the sum is obtained in a time which is approximately proportional to the logarithm of the number length.

Finally, we mention here the *anticipatory-carry* or *carry-skip* technique first proposed by Babbage,[8] long used in decimal desk and punched-card machines, and lately revived for use in binary digital computers.[9,10] The technique, whose development forms the main topic of the present paper, will be further described in subsequent sections.

In briefly summarizing known methods of speeding up carry-propagation, we have not mentioned the *carry storage* philosophy originally due to Burks, *et al.*,[2] and lately developed by the computer group at the University of Illinois.[11,12] This method achieves a speed-up of the arithmetic process as a whole rather than only a minimization of the propagation effect. Moreover, it is compatible with the other methods outlined and its characteristics are therefore not considered relevant to the present study.

II. Carry Skip

In their contribution to the IEE Discussion Meeting on New Digital Computer Techniques, Morgan and Jarvis[10] describe a binary skip circuit. The technique is based on the detection and by-passing of those stages of a parallel binary adder in which, during a given addition $(X+Y)$, there exists the condition for carry-propagation. That is, the carry-signal is enabled to by-pass those stages of the carry circuits for which

$$x_i \neq y_i. \tag{1}$$

An alternative criterion which does not differentiate between propagated and generated carries but which is more efficient in the skip circuit is

$$x_i \cup y_i. \tag{2}$$

The circuit described by Morgan and Jarvis and outlined in block diagram form in Fig. 1 divides the adder into fixed groups, each of six stages. The carry-signal appearing at the input of any group for which condition (2) is satisfied for each stage of the group, is transmitted to the next group through a special *skip gate*. At the same time the carry is also permitted to propagate within the group to permit determination of the various sum bits.

It is clear that the division into groups of 6 bits (for a word-length of 54 bits) which in the above circuit appears to have been based on circuit considerations, need not prove optimum from the point of view of logical efficiency. Furthermore, in the circuit described, propagation chains of up to ten stages can still occur without excitation of any skip gates. The following fundamental questions may therefore be posed:

1) In the Morgan and Jarvis version of the carry-skip technique, what is the optimum group division for equal-sized groups, for binary numbers of length $(m+1)$ bits?

2) Can the circuit be further speeded up by a division into unequal groups?

3) Is it advantageous and economic to provide additional skip gates within the groups or between the groups or both?

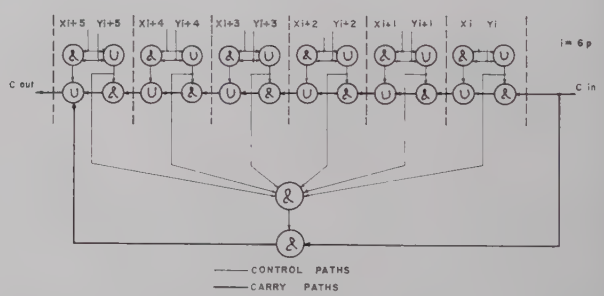


Fig. 1—The simple skip circuit according to Morgan and Jarvis.

III. Worst-Case Design

Before further developing the skip technique, one fundamental point must be clarified. In all addition systems except *carry-detection*, the circuit must be based on "worst-case design"; that is, the duration of all additions are equal, being determined by the time required to add two numbers whose bit patterns give rise to maximum time of propagation. The result of an addition can never be used elsewhere in the machine before this time has elapsed. Thus any components which do not contribute to a minimization of the worst-case time but merely serve to make certain (or all) of the sum bits available earlier for other bit patterns, are redundant.[13] The actual time allowed for carry-propagation will normally be equal to the worst time plus a safety margin which may add as much as 50 to 100 per cent to the latter. This margin is not further considered or mentioned in the present analysis.

[8] M. V. Wilkes, "Automatic Digital Calculators," Methuen & Co., London, Eng., pp. 13–15; 1958.

[9] "Specialist Discussion Meeting on New Digital Computer Techniques—Special Aspects of Logical Design," *Proc. IEE*, pt. B, vol. 106, pp. 462–469; September, 1959.

[10] C. P. Morgan and D. B. Jarvis, "Transistor logic using current switching and routing techniques and its application to a fast-carry propagation adder," *Proc. IEE*, pt. B, vol. 106, pp. 467–468; September, 1959.

[11] "On the Design of a Very High Speed Digital Computer," Digital Computer Lab., University of Illinois, Urbana, Rept. No. 80; October, 1957.

[12] G. Metze and J. E. Robertson, "Elimination of carry propagation in digital computers," *Proc. Internatl. Conf. on Information Processing*, Paris, France, pp. 389–396; June 15–20, 1959.

[13] M. Lehman and N. Burla, "A note on the simultaneous carry-generation system for high-speed adders," IRE Trans. on Electronic Computers, vol. EC-9, p. 510; December, 1960.

IV. Operation Time

In the same way as the speed of the *simultaneous-carry* adder depends on the number of phases used, the speed of a skip adder depends not only on the circuit elements and electronic circuits but also on the logical structure of the propagation network. The designer must determine the addition time which he desires to achieve taking into consideration economic factors and the general system design. After this initial decision he may then proceed to the development of a practical skip circuit. The ultimate upper limit to the speed is that which is obtained when all possible skip-paths are provided for. It is interesting to note that in this latter case the circuit *degenerates* into an instantaneous carry circuit which, in principle, is identical with a two-phase, nonredundant version of the *simultaneous-carry* system.

V. The Optimum Size of Equal Groups

Consider an adder of length $(m+1)$ bits, which is to be divided into k skip-groups each of n bits so that:

$$nk = m + 1. \tag{3}$$

Let the basic time unit (t.u.) be of the order of magnitude of the time required to propagate a signal (carry) through an AND-OR gate combination. In the present, approximate, analysis it is not possible to consider the delay arising from passage through a gate a function of gate size. This approach is justified by the fact that in general the skip-gates are primed or preset, with only that element involved in the actual propagation path being active during the propagation process. It is also not necessary to consider the type of elements (*e.g.*, diodes, transistors, ferrites) being used since in any one design the majority at least of the elements will be of the same type.

It is clear that the worst case arises where a carry is generated in the least significant bit-stage and after a full-length propagation is terminated in the most significant stage; that is for addend and augend

$$X = 2^p \sum_{i=0}^{m} x_i 2^i, \quad Y = 2^p \sum_{i=0}^{m} y_i 2^i,$$

the following conditions exist:

$$x_0 \,\&\, y_0 \tag{4}$$

$$x_i \cup y_i \tag{5}$$

$$\bar{x}_m \,\&\, \bar{y}_m. \tag{6}$$

In those circumstances the time required for propagation is

$$\begin{aligned} T &= [1 + (n-1) + (k-2) + (n-1)]\text{t.u.} \\ &= (2n + k - 3)\text{t.u.} \\ &= \left(2n + \frac{m+1}{n} - 3\right)\text{t.u.} \end{aligned} \tag{7}$$

T is measured from the moment when the operands are presented to the carry circuit to the time when propagation must be completed and the sum may be determined. It is required to minimize T with respect to n. Differentiating and applying the standard condition for a minimum gives

$$\frac{dT}{dn} = 2 - \frac{m+1}{n^2} = 0, \tag{8}$$

that is,

$$n = \sqrt{\frac{m+1}{2}} \tag{9}$$

or

$$k = 2n. \tag{10}$$

The general trends of k and n as functions of word-lengths are shown in Fig. 2. In any practical case it is simple to determine the optimum integral values, though their final selection will also be determined by other considerations. Fig. 3 shows a plot of relative

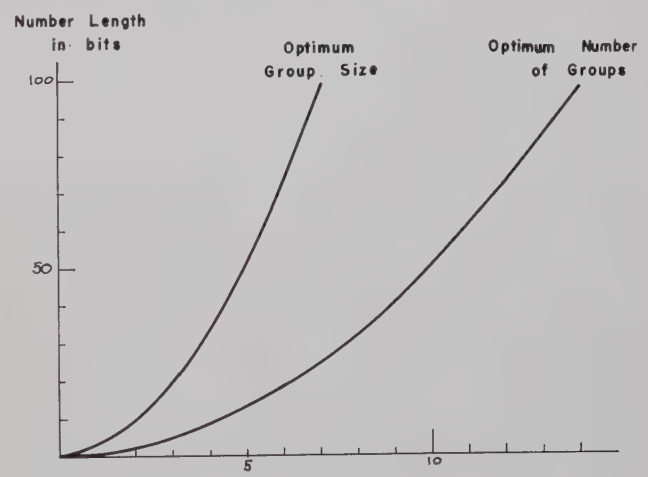


Fig. 2—The optimum group size and the optimum number of groups as a function of word length—for equal groups.

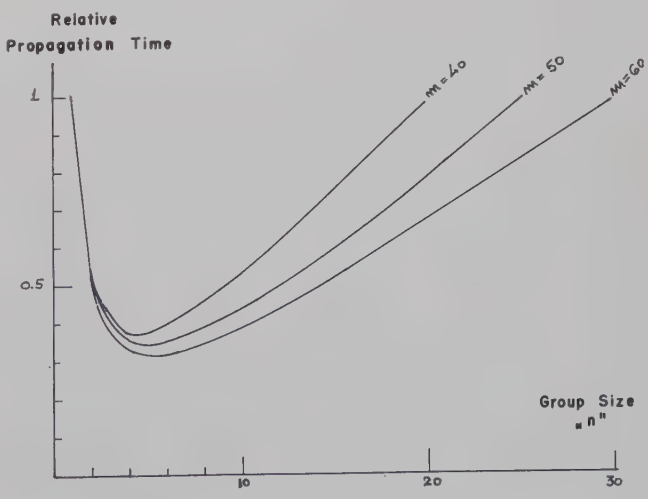


Fig. 3—The relative values of maximum propagation time for deviations from the optimum group size—for equal groups.

propagation time for deviation from optimum size groups, for various number lengths, normalization having been obtained by dividing the time T as defined by (7), by the number-length m.

VI. Nonequal Groups

In Section V, a technique providing skip paths over equal-sized groups was discussed and optimized. The design was based on a worst case in which a full-length carry excluded only the least and most significant bit pairs. These, of course, generate and terminate the propagation respectively. For the more general case, in which one or more propagation chains are generated at each group up to the middle group (or pair), and then decrease in the same way so that the dependence of the propagation time on the chain-length is almost eliminated. A structural change of this type will always achieve a reduction in propagation time as compared with division into equal groups without any change in the amount of equipment required. Table I shows how a 60-bit adder, for which the optimum equal-group division is ten groups each of 6 bits (or twelve groups each of 5 bits), may be divided to give a speed-up of 20 per cent without any increase in the amount of equipment provided only that any limit on basic gate-size is not exceeded.

TABLE I
Possible Group Sizes for a 60-Bit Adder

	Maximum Propagation Time in t.u.	Group Identification and Size									
		j	i	h	g	f	e	d	c	b	a
Equal Groups	19	6	6	6	6	6	6	6	6	6	6
Nonequal Groups	15	4	5	6	7	8	8	7	6	5	4

internal points of the carry network, it is easily verified that a time less than T is required for completion of the propagation process.

This fact implies redundancy in the circuit, since the final carry distribution is completed before it may be used, unless a completion-of-carry detection-circuit is incorporated. This redundancy may be reduced, in units not using *ones complements* representation of negative numbers with end-around-carry, by increasing the relative size of inner groups in relation to those of the groups at the extremes of the circuit. Consider a parallel $(m+1)$ bit adder initially divided, as above, into k, n-bit groups. Suppose the size of the least significant group g_0 is now reduced by one to $(n-1)$ bits, the size of the second group g_1 is unchanged (as a result of the addition of a bit at its least significant end and the removal of a bit from its most significant end) and the size of the third group g_2 is increased by one to $(n+1)$ bits. For the worst case as previously defined, the over-all propagation time is now $(T-1)$ t.u. where T is given by (7) since the number of skip gates is unchanged but the number of gate-pairs in the unskipped group g_0 is reduced by one. The duration of a propagation chain generated in the least significant end of g_1 or g_2 and propagated to the most significant end of the numbers is, for the modified design, also $(T-1)$ t.u., since, as the number of skip gates passed is reduced, the number of bit-gates in the originating group has been increased in such a way as to keep the total number of gates in the propagation path constant. This process of redistribution of circuits to obtain unequal groups, may be continued and the optimum division is clearly obtained when group sizes increase by one stage for

VII. Further Development

The skip techniques described so far are essentially primitive in that only a few of the many possible propagation chains may be by-passed. The limitations of the earlier approach may be overcome by providing a nest of additional internal skips for each group or a net of intergroup skips. Such circuits have previously been described by Burla[14] and by MacSorley,[15] but for a complete picture are presented again in Figs. 4 and 5. The former indicates how additional skip circuits within a group can produce a nest which propagates any carry entering or generated within the group in, say, only one time unit. Fig. 5 on the other hand shows how groups of groups may be by-passed. A further alternative not illustrated would interlace several sets of groups so that any carry wherever generated could immediately skip, and the need for propagation in each group would be largely eliminated.

These various approaches are of course not exclusive and their joint logical limit, in which every logical skip is provided for, has already been mentioned. Such an instantaneous adder would, for $(m+1)$-bit operands, require

$$8(m+1) + \sum_{i=1}^{m-1} (i+3)(m-i) = \tfrac{1}{6}(m^3 + 9m^2 + 38m + 48) \quad (11)$$

[14] N. Burla, "Some Logical Problems in the Design of a High Speed Adder for Parallel Binary Digital Computers," M.S. Thesis, The Technion, Israel Inst. of Tech., Haifa; May, 1960.

[15] O. L. MacSorley, "High-speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67–91; January, 1961.

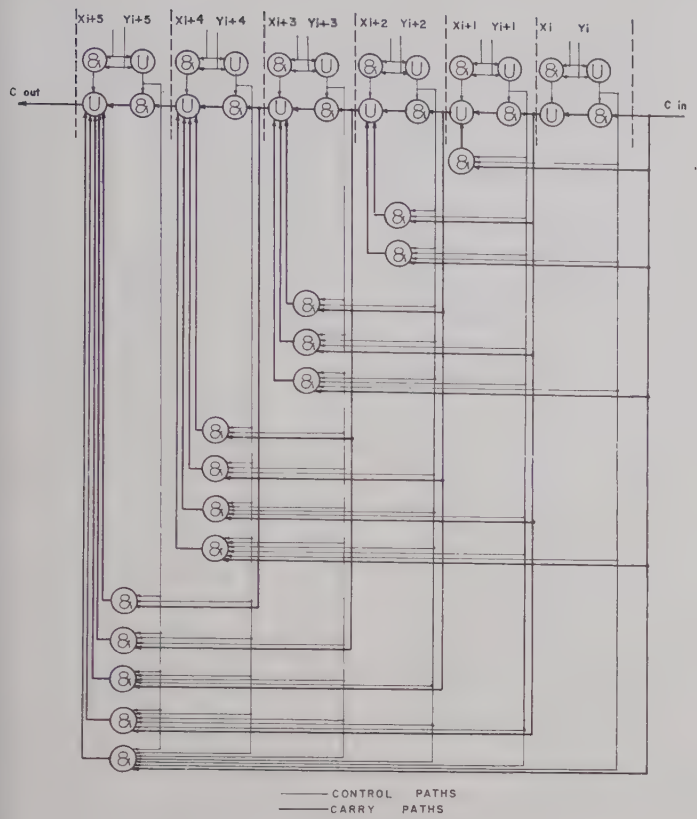


Fig. 4—Example of a complete nest of internal skips for propagation over a 6-bit group in one time unit.

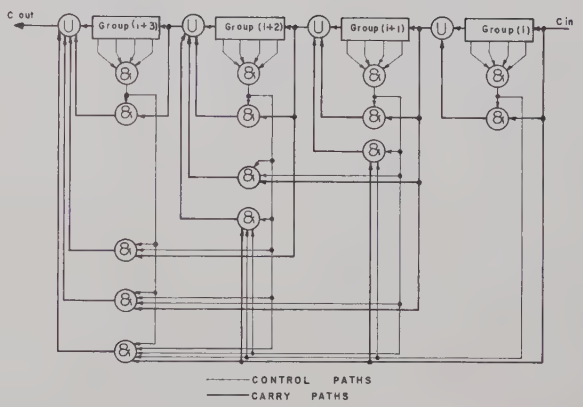


Fig. 5—An example of a complete intergroup skip-net for 4-bit groups.

logical elements (*e.g.*, diodes). The largest gate in the resultant network would require $(m+1)$ inputs. This arrangement appears impractical for general-purpose computers. In any practical circuit the designer will normally be satisfied to adopt a less ambitious scheme, based on the practical limitation of maximum gate-size and the arbitrary speed which it is desired to achieve and which can never be less than 2 t.u. as previously defined.

The first step in designing a carry network is the fixing of the above two factors. For a carry network for which the target speed is $(p+1)$ t.u. and which is to utilize gates having a maximum of q inputs, the basic task of the designer is to divide the carry network into groups of bit stages or into groups of subgroups of bit stages, each of which is to be by-passed by a skip gate. The following heuristic principles may guide the general design procedure.

1) The design is based on a division of the $m+1$-bit adder, into p skip-groups. This follows from the required propagation time of p t.u., since one time-unit is always required for the initial insertion of addend and augend bits and for the formation of the elementary carry-functions. These p groups need not necessarily each contain the same number of bit stages, but at this point in the design procedure they are assumed to be of approximately equal size.

2) A further division into subgroups is derived where the number of stages n in each subgroup must be less than q (maximum gate size).

No precise set of rules enabling the determination of the boundaries between subgroups and groups have been established, since in practice the details of the design are functions of many factors falling outside the scope of this paper.

The following further principles indicate however the general trend of the design technique, which is based on examination of all possible gates:

a) As previously noted the *final* carry bits in each stage from which the sum bits are subsequently computed need appear nowhere before conclusion of $(p+1)$ t.u. from the moment the operands are applied to the network. Thus, skip gates not required to ensure that all carries are available by this time, are redundant, and should not be inserted. It is of interest to note that this principle applies not only to skip but equally to other high-speed carry circuits.[13]

b) In general, where the alternative exists, it is preferable to make subgroups small, increasing the number of intergroup skips and reducing the number of internal skips. From a) and b) it follows that, if p is small, n should be chosen less than p in which case internal skip may be completely eliminated.

c) For units not using *ones complements* representation of negative numbers, the relative sizes of all groups need not be the same. For the optimum design, the middle group may be larger, and groups nearer the least and most significant ends, smaller. This follows from a) and is similar in principle to the technique described in Section VI.

VIII. Examples

The principles governing the design of carry-skip circuits can best be mastered by undertaking an actual design.

As previously mentioned, details of any practical design are subject to a number of arbitrary decisions. These are, amongst other factors, functions of number length, desired speed and circuit components to be used. We present here two proposals for systems, one simple, one complex, in the form of logical schematics which should be largely self-explanatory. The following notes emphasizing the most important aspects of the design may however prove of interest.

Fig. 6(a) and 6(b) illustrate a 48-bit adder operating in 12 t.u. The circuit is *simple* in that internal skips are not used, there being only twelve subgroups of 4-bit stages each, further combined into five groups of differing sizes. It calls for a maximum gate-size of five inputs as in Fig. 6(a) (group 3). This five-input skip gate could be reduced by one element through the provision of a two-element skip gate, controlled from a four-element skip-detection gate. Use of the latter technique would further reduce propagation time by slightly reducing the basic time unit.

The main point worthy of stress is the essential economy of the circuit. It requires an average of only ten gating elements per bit stage as compared with the eight required by a conventional circuit. For this trivial addition of equipment it achieves an actual reduction in addition time to some twenty-five per cent of that of a conventional circuit since it requires 12 t.u. for completion of propagation as compared with the 48 t.u. required by the conventional circuit.

The second example illustrated in Fig. 7(a)–7(d) is far more complex and is intended to illustrate all aspects of the skip technique.

The 54-bit adder [Fig. 7(a)] is ready for determination of the sum after four t.u. ($p=3$). This speed is achieved through division into three groups totaling nine subgroups. Of the latter, the middle seven comprise a single group. The subgroups are all of equal size and it follows that in order to complete propagation in the required time, both internal and intersubgroup skip gates are required.

In the least significant group [Fig. 7(b), first subgroup] it is required to expedite *emergent* carries so that skip gates feed the generated carries from the various internal points to the exit point of the group.

For the most significant group [Fig. 7(c), ninth subgroup] on the other hand, it is necessary to expedite the distribution of *incident* carries. Thus skip gates fed from the common entry to the group have outputs connected to the various bit stages. For the middle group, Fig. 7(a) (for groups and intersubgroups) and Fig. 7(d) (for internal skips) show how sets of gates (not unique) of incident, emergent and internal-skip types provide for completion of carry under all conditions in the required time of three t.u.

The total element count for this circuit shows an average of order eighteen logical elements per bit stage, an increase of 125 per cent as compared with the conventional carry circuit. The time that has to be permitted for carry-propagation on the other hand is reduced to 7.3 per cent of its conventional value.

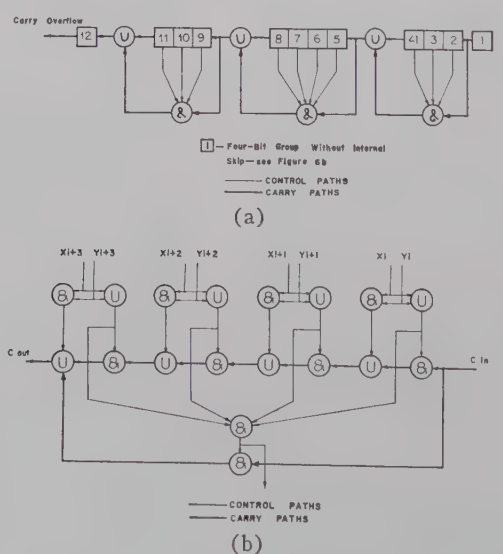
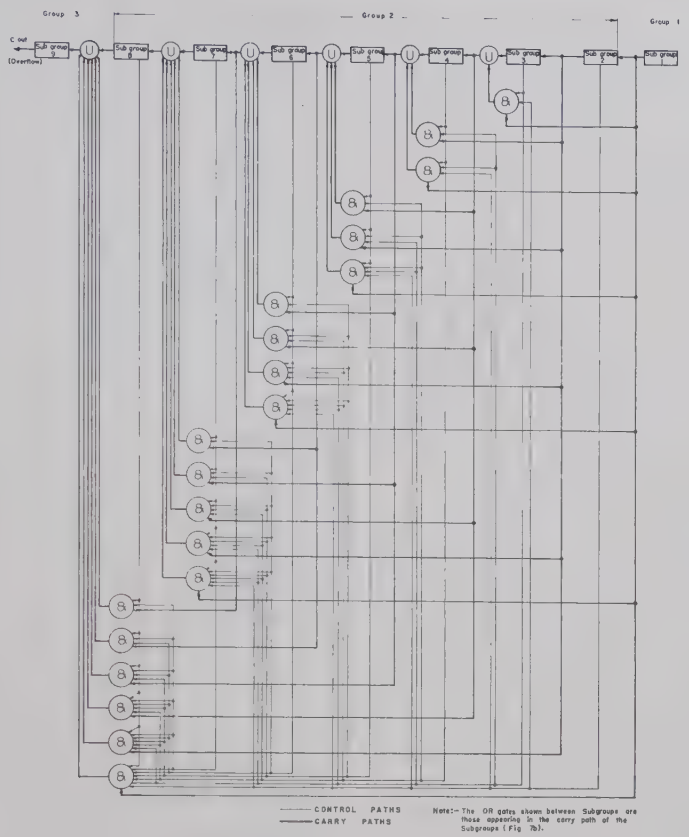
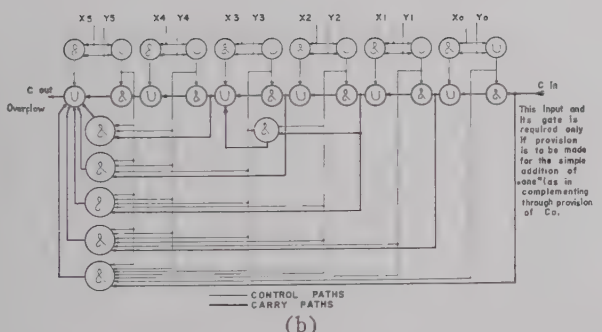


Fig. 6—Example One

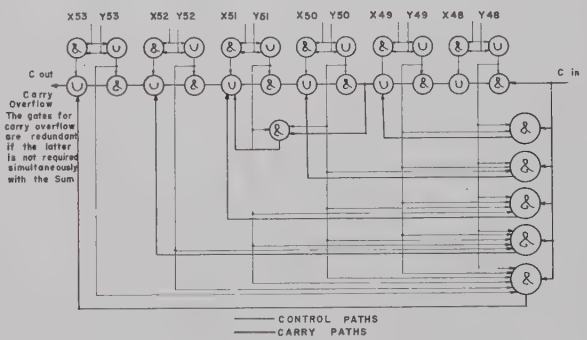
(a) A simple skip adder, the intersubgroup skip.
(b) The internal structure of the subgroup



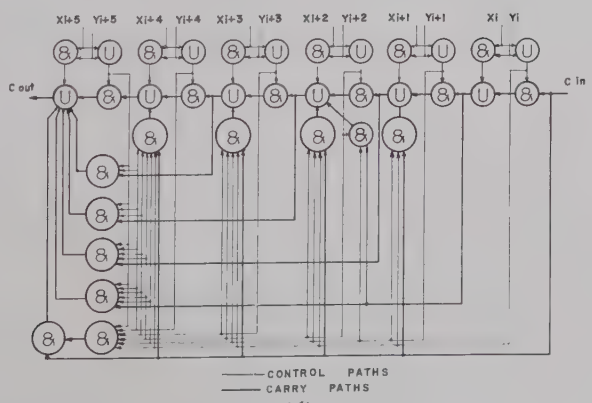
(a)



(b)



(c)



(d)

Fig. 7—Example Two

(a) The intergroup and intersubgroup networks.
(b) The first subgroup—by-pass mainly of emergent carries.
(c) The ninth subgroup—by-pass mainly of incident carries.
(d) The intermediate groups requiring gates for incident, internal and emergent carries.

IX. Conclusion

A comparison has recently been made[14] between all those methods of carry speedup described in the present paper excepting those of Nadler[6] and Sklansky.[7] The comparison was based on a criterion

$$Q = \frac{t}{t_0} \cdot \frac{D}{D_0} \cdot 100 \tag{13}$$

where

t = the operation time of the network under consideration

t_0 = the operation time of an equal length, conventional, parallel, carry circuit.

D = the total number of elements in the carry network under consideration

D_0 = the number of elements in a conventional circuit and, for example, equals $8(m+1)$ where diodes are used and where $m+1$ is the number length.

This comparison has now been extended to include also the pyramid and conditional sum circuits and a paper presenting the complete result is in preparation. Table II summarizes the results of this further study for circuits based on diode elements.

Though the present study will not justify or discuss these results, the following comments should be added.

The "exclusive OR" circuit has been included in the table though numerical assessment of its relative standing is not, here, practical. The circuit is based on the use of high-speed transistors and special circuit techniques and it could be usefully compared only with other fully designed and operative circuits. The same techniques could, of course, be applied, at some extra expense to many of the other circuits.

The figure of 25.5 for the number of diodes in the *conditional sum* adder includes two extra for each M circuit, replacing the Invertor I in Fig. 2(e) of Sklansky.[7] No account had here been taken of the need in this, or the pyramid circuit, for temporary storage or synchronizing elements.

TABLE II

A Comparison of Various Carry-Network Parameters

Method	t	$t/t_0 \cdot 100$	$D/m+1$	Q
Conventional	54	100	8	100
"Exclusive OR" (Kilburn, *et al.*)	$\ll 27$	$\ll 50$	>13	—
Simple Skip (Morgan, *et al.*)	18	34	9.5	40
Conditional Sum (Sklansky)	5.5	10	25.5	32
Skip—Example One (Lehman, *et al.*	12	22	10	28
Carry-Detection (Gilchrist, *et al.*)	$5.6 < t < 6.6$ (average)	10.4	19	25
Pyramid (Nadler)	4	7.4	25.2	23
Simultaneous (Weinberger, *et al.*)	4	7.4	23	21.2
Skip—Example Two (Lehman, *et al.*)	4	7.4	18.5	17

Since each stage of the pyramid circuit consists only of AND gates, a fairer estimate of the delay per stage would be of the order of $\frac{1}{2}$ t.u. In the last two stages these gates are very large so that the delay is likely to approach 1 t.u. Hence the total time for the circuit has been estimated at 4 t.u.

The data for three examples of the skip technique are included in the table. These appear to indicate that, in general, for a given expenditure the skip technique leads to the fastest adder. Conversely, for a given speed, the skip-adder tends to be the most economical. An *accurate* estimate of relative costs and efficiencies, however, requires the comparison of *the circuits of operational equipment* since the present study (or any equivalent) must be based on a number of assumptions. Thus Table II should merely be regarded as indicating the efficiency of the skip technique, suggesting that further detailed circuit design of various alternative schemes appears well-justified.

Some Properties of Binary Counters with Feedback*

ROBERT C. BRIGHAM†, ASSOCIATE MEMBER, IRE

Summary—**Properties of binary counters which have feedback connections are discussed. Various methods of determining the cyclic period have been reported, as well as techniques for synthesizing feedback connections to obtain a specific period. This paper presents a somewhat different approach to the period determination. It also includes a discussion of a simple method for determining the numerical values eliminated from the normal counting sequence by feedback connections. Several illustrative examples are given.**

Introduction

FIG. 1 is a symbolic representation of an n-bit binary counter which can hold the binary equivalent of any of the decimal integers 0, 1, $\cdots$, 2^n-1. Each numbered block, known as a "stage" of the counter, is a bistable device, and an input pulse to it changes its binary value. Each stage produces an output pulse when its value changes from one to zero. The output pulse of stage j serves as an input pulse to stage $j+1$. The input pulses to stage 1 are the source pulses which are to be counted. The total number of input pulses required to cycle the counter completely is 2^n. We shall refer to this as the "count" C. Thus, if $n=3$, the counting sequence will be 000, 100, 010, 110, 001, 101, 011, 111, 000, $\cdots$, having a period of $2^3=8$.

The count can be made smaller than 2^n by introducing feedback connections. Feedback occurs when the output pulse of one stage serves, after appropriate delay, as an input pulse to itself, any preceding stage, or any combination of itself and preceding stages. We shall represent feedback connections symbolically by arrows issuing from a stage, as shown in Fig. 2. Here the output pulse of stage $n-1$, in addition to becoming an input pulse to stage n, serves as an input pulse to both stages 1 and 3.

It has been shown[1] that feedback can cause the count to be any of the integral values 1, 2, $\cdots$, 2^n-1. Various methods of determining the count, as well as of synthesizing feedback connections to obtain a specific count, have been reported.[2,3] This paper presents a somewhat different approach to the count determination. It also describes other properties of binary counters, including a simple method for determining the numerical values eliminated from the normal counting sequence by feedback connections.

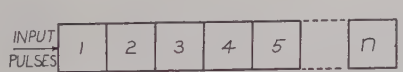


Fig. 1—Symbolic representation of n-bit binary counter.

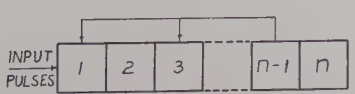


Fig. 2—Simple form of feedback connection.

Basic Feedback Paths

A basic feedback path is defined to be one which can be designated as $j(A)$, where j is the number of the stage, known as the controlling stage, whose output pulse produces the feedback, and A is the decimal equivalent of the positive binary integer obtained when each stage receiving feedback is considered as one and all others are considered as zero. It can be seen that $A \leq 2^j-1$. Thus, the connection of Fig. 3 is designated 4(5). Fig. 4, for the moment considering the dotted connection as a solid one, is composed of the two basic connections 4(3) and 7(57).

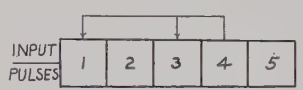


Fig. 3—A 4(5) feedback connection.

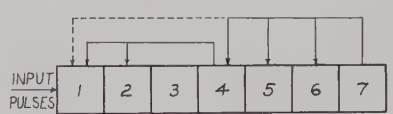


Fig. 4—The two feedback connections 4(3) and 7(57).

The expression $j(A)$ can represent any of the 2^j-1 basic feedback paths $j(1)$, $j(2)$, $\cdots$, $j(2^j-1)$. The number of different possible basic paths in an n-stage counter is then

$$\sum_{j=1}^{n}(2^j-1) = (2^1+2^2+\cdots+2^n)-n$$
$$= (2^{n+1}-2)-n = 2^{n+1}-(n+2).$$

If feedback is permitted only to preceding stages, the number of different paths is

$$\sum_{j=2}^{n}(2^{j-1}-1) = (2^1+2^2+\cdots+2^{n-1})-(n-1)$$
$$= (2^n-2)-(n-1) = 2^n-(n+1).$$

* Received by the PGEC, February 2, 1961.
† Martin Co., Orlando, Fla.
[1] H. Lifschutz, *Phys. Rev.*, vol. 57, p. 243; 1940.
[2] B. R. Gossick, "Predetermined electronic counter," PROC. IRE, vol. 37, p. 813; July, 1949.
[3] G. F. Montgomery, "Cascaded binary counter with feedback," *J. Appl. Phys.*, vol. 22, pp. 780–789; June, 1951.

A 5-stage counter will have 57 different basic feedback paths under the former conditions and 26 under the latter.

The Count Required to Cycle

Consider a single basic feedback path $j(A)$. With no feedback, all of the stages 1, 2, $\cdots$, j are set to zero when stage j produces an output pulse. With feedback, however, this pulse places the number A in the first j positions of the counter. Thus, A counts are eliminated each time stage j changes its value from one to zero. This latter event occurs 2^{n-j} times in a complete cycle so that the count C required to cycle completely is

$$C = 2^n - A \cdot 2^{n-j}.$$

The 4(5) path of Fig. 3, therefore, causes cycling every

$$C = 2^5 - 5.2^{5-4} = 22 \text{ counts.}$$

Redundancy is a possibility when a counter has the two basic feedback paths $j(A)$ and $k(B)$, $k>j$. Redundancy is defined to exist whenever stages k and j provide feedback pulses to the same stages. Examination shows that stage j produces an output pulse every time that stage k does. Therefore, any path to a common stage (one which receives feedback input pulses from both stages j and k) which is controlled by stage k can be eliminated for purposes of analysis. The result is a new basic path $k(B')$ controlled by stage k. The following results apply only if this elimination is done. This does not imply eliminating such connections from the physical counter, although they serve no purpose there. Redundant connections will be indicated by dotted lines in the remainder of this paper. It can be seen that $B' = B_1' + B_2'$ with $B_1' = m \cdot 2^j$ (m is an integer such that $m \cdot 2^j \leq B' < (m+1)2^j$). After redundancies are removed, each of the 2^{n-k} times that stage k produces an output results in the elimination of B' counts. Stage j, which normally produces 2^{n-j} outputs in a complete cycle, has this number reduced by $B_1'/2^j$ each of the 2^{n-k} times that stage k produces an output. Therefore,

$$C = 2^n - A[2^{n-j} - 2^{-j}(B_1' \cdot 2^{n-k})] - B' \cdot 2^{n-k} \text{ counts.}$$

In Fig. 4, $B' = 56$, $B_1' = 48 = 3 \cdot 2^4$, $B_2' = 8$ and

$$C = 2^7 - 3[2^{7-4} - 2^{-4}(48 \cdot 2^{7-7})] - 56 \cdot 2^{7-7} = 57 \text{ counts.}$$

Two special cases exist:

$$B_1' = 0: \quad C = 2^n - A \cdot 2^{n-j} - B' \cdot 2^{n-k} \text{ counts;}$$

$$B_2' = 0: \quad C = 2^n - A[2^{n-j} - 2^{-j}(B' \cdot 2^{n-k})] - B' \cdot 2^{n-k} \text{ counts.}$$

The above results can be extended to the general case when m different basic feedback paths are employed. The following rules will yield the correct results.

1) Eliminate all redundant connections (for analytic purposes only).
2) Determine the number of outputs produced by each controlling stage. For controlling stage j, this number is equal to

$$N_j = 2^{n-j} - 2^{-j} \sum_k B_1' \cdot N_k.$$

B_1' is obtained by considering the feedback controlled by stage j individually with that controlled by each stage k for all $k>j$. The result is the general two-feedback connection case discussed above. N_k is the number of outputs produced by stage k and will always be available if this process is initiated with the rightmost controlling stage and worked to the left.
3) Determine the counts lost due to feedback resulting from each controlling stage. This is accomplished by multiplying each of the results of step 2 by the number defining the stages receiving feedback for that particular controlling stage, with no redundant connections.
4) The cyclic count C is 2^n minus the sum of the results of step 3.

As an illustration, consider Fig. 5.

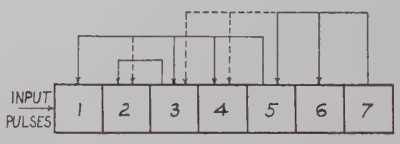


Fig. 5—3(2), 5(15), and 7(60) feedback connections.

Controlling Stage	N_k	*Counts Lost*
7	$2^{7-7} = 1$	$1 \cdot 48 = 48$
5	$2^{7-5} - 2^{-5}(32 \cdot 1) = 3$	$3 \cdot 13 = 39$
3	$2^{7-3} - 2^{-3}[48 \cdot 1 + 8 \cdot 3] = 7$	$7 \cdot 2 = 14$

$$C = 2^7 - 48 - 39 - 14 = 27 \text{ counts.}$$

Replacing Multiple Basic Paths by a Single Path

If C is known, the counts eliminated E can be determined by noting that $C = 2^n - E$. This same count can be achieved by a single basic feedback connection $k(B)$ such that $E = B \cdot 2^{n-k}$. Proper values of B and k can always be selected because $E \leq 2^n - 1$ and k may be set equal to n. This basic path may not be the one with the least number of connections.[3]

If a counter contains the two basic feedback connections $j(A)$ and $k(B)$, $k>j$, the same count can always be obtained by a single feedback path controlled by stage k. It will be recalled that the general formula for the count for two connections is

$$C = 2^n - A[2^{n-j} - 2^{-j}(B_1' \cdot 2^{n-k})] - B' \cdot 2^{n-k}$$
$$= 2^n - 2^{n-k}(B' + A \cdot 2^{k-j} - AB_1' \cdot 2^{-j}).$$

This is the correct form for the single basic feedback path $k(B' + A \cdot 2^{k-j} - AB_1' \cdot 2^{-j})$, provided that

$B'+A\cdot 2^{k-i}-AB_1'\cdot 2^{-i}\leq 2^k-1$. To show this latter point, it will be remembered that $B_1'=m\cdot 2^i$ where m is an integer such that $m\leq 2^{k-i}-1$. In addition, after all redundancies are removed, $B'\leq(m+1)2^i-1-A$, since $A+B'\leq 2^k-1$. Thus,

$$\begin{aligned} B' + A\cdot 2^{k-i} - AB_1'\cdot 2^{-i} &\leq (m+1)2^i - 1 - A + A\cdot 2^{k-i} - Am \\ &\leq m(2^i - A) + 2^i - 1 - A + A\cdot 2^{k-i} \\ &\leq (2^{k-i} - 1)(2^i - A) + 2^i - 1 - A + A\cdot 2^{k-i} \\ &\leq 2^k - 2^i - A\cdot 2^{k-i} + A + 2^i - 1 - A + A\cdot 2^{k-i} \\ &\leq 2^k - 1 \end{aligned}$$

which was to be shown.

Determination of the Numerical Values Eliminated by Feedback Connections

Consider a $j(A)$ basic feedback connection. If the counter holds the number 2^n-1, the next pulse results in the counter containing the number A, rather than zero, as would be the case in the absence of feedback. Thus, the values $0, 1, \cdots, A-1$ are eliminated. The count will then continue normally from A to 2^{i-1}. The next pulse would normally place the number 2^i in the counter. However, because of the feedback connection, the counter will actually hold 2^i+A after the next pulse, that is, when stage j produces its next output pulse. Thus, the numbers $2^i, 2^i+1, \cdots, 2^i+A-1$, are also eliminated. Similarly, a set of A consecutive numbers is eliminated each of the 2^{n-i} times stage j produces an output, and the sets are separated by 2^i units. If α represents the set of numbers $0, 1, \cdots, A-1$, the numbers eliminated in a complete cycle are

$$\alpha,\ \alpha+2^i,\ \alpha+2\cdot 2^i,\ \alpha+3\cdot 2^i,\ \cdots,\ \alpha+(2^{n-i}-1)2^i.$$

The 4(5) connection of Fig. 3 is easily seen to eliminate the numbers 0–4 and 16–20.

It is now advisable to define the "logical or" of controlling stage k as the basic feedback path $k(B'')$. B'' is the decimal equivalent of the binary integer obtained when each stage receiving feedback, if it is controlled by stage k or any controlling stages to the *left* of stage k, is considered as one and all other stages are considered as zero. The two "logical or" paths of Fig. 4 are 4(3) and 7(59), and the three of Fig. 5 are 3(2), 5(15), and 7(63).

Rules for the determination of the numbers eliminated by any combination of basic feedback connections can now be stated.

1) Obtain the "logical or" for each controlling stage.
2) For each result of step 1, determine a set of numbers in the manner indicated in the first paragraph of this section.
3) Obtain a list of all the eliminated numbers by combining the results of step 2 in such a way that a number is considered eliminated if it appears anywhere in these results.

As an illustration, consider Fig. 4:

"Logical or"	*Numbers Eliminated*
4(3)	0–2, 16–18, 32–34, 48–50, 64–66, 80–82, 96–98, 112–114
7(59)	0–58.

The eliminated numbers are 0–58, 64–66, 80–82, 96–98, 112–114. For Fig. 5:

"Logical or"	*Numbers Eliminated*
3(2)	0–1, 8–9, 16–17, 24–25, 32–33, 40–41, 48–49, 56–57, 64–65, 72–73, 80–81, 88–89, 96–97, 104–105, 112–113, 120–121
5(15)	0–14, 32–46, 64–78, 96–110
7(63)	0–62.

The eliminated numbers are 0–62, 64–78, 80–81, 88–89, 96–110, 112–113, 120–121.

It may be noted that if a combination of feedback connections eliminates the set of numbers α, this combination in conjunction with additional connections eliminates a set of numbers β such that β contains α.

Acknowledgment

The author wishes to express his gratitude to C. E. McGinnis and H. McGuire for encouraging the preparation of this paper and providing time for it.

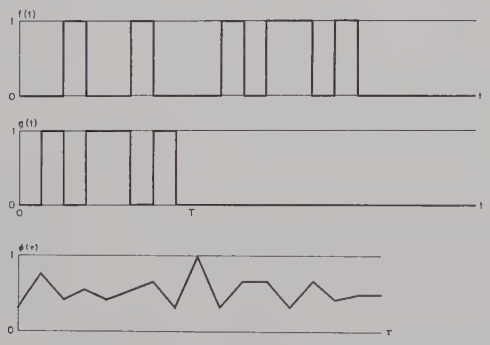
A Magnetostrictive Delay-Line Shift Register*

LEE E. HARGRAVE, JR.†, MEMBER, IRE

Summary—**A brief theory of correlation, as applied to the recognition of a specific pattern in a binary signal, is presented. The digital shift register, which lends itself well to the practical application of the theory of correlation, is discussed. By digitizing the binary input signal and applying it to a closed-loop delay line, a time-compressed version of the input signal circulates on the line, and the delay line can perform the same function as the digital shift register, with possible advantages of size and cost over the shift register. Sample parameters are postulated for a shift register, and a shift register employing a magnetostrictive delay line is designed. The operation of the designed delay line shift register is then discussed in detail.**

I. A General Discussion of Correlation

GIVEN a binary signal in noise, the problem of recognizing a known pattern therein lends itself readily to correlation techniques. The problem is illustrated by an example in Fig. 1, where the binary signal is $f(t)$. For simplicity, the two permissible signal states are denoted one (mark) and zero (space). The task is to recognize a known pattern whenever it occurs in $f(t)$. This specific pattern, or word, is designated $g(t)$, the word length being T seconds. In the example of Fig. 1, the simple word pattern $g(t)$ obviously occurs in the signal $f(t)$.



$$\phi(\tau) = \frac{1}{T}\int_0^T f(t+\tau)\, g(t)\, dt$$

Fig. 1—Binary correlation.

The correlation function $\phi(\tau)$ is given by

$$\phi(\tau) = \frac{1}{T}\int_0^T f(t+\tau)g(t)dt.$$

In the example, whenever $\phi(\tau)$ assumes its maximum value of unity, a word is present in $f(t)$ from $t=\tau$ to $t=\tau+T$. At any given time, the value of the correlation function will lie between zero and one, assuming its minimum value of zero when the inverse of the word $g(t)$ appears in $f(t)$.

* Received by the PGEC, November 21, 1960.
† Sanders Associates, Inc., Nashua, N. H

The explanation of the theory of correlation simplifies considerably when the signal is binary. Whenever $f(t+\tau)$ and $g(t)$ are at the same state, their product is one; whenever their states are different, a zero product results. Since the integral is only taken from $t=0$ to $t=T$, only the product in this interval is of interest. The effect is that of a template of the word structure from $t=0$ to $t=T$. As τ increases, the binary signal $f(t+\tau)$ slides through the template from right to left, yielding an instantaneous product of the signal and the template. The correlation function $\phi(\tau)$ is simply the integral of the product over the template normalized to a maximum of unity. In other words, the correlation function $\phi(\tau)$ indicates that fraction of the signal which agrees with the template at any time τ. In Fig. 1, $\phi(\tau)$ is plotted for a sample $f(t)$ and $g(t)$; note the peak of $\phi(\tau)$ at the leading edge of the word pattern in $f(t)$.

In practice, a valid word in the binary signal could be corrupted by the effects of noise, resulting in imperfect peaking of the correlation function. To combat this and improve the probability of recognition, a recognition threshold would be assigned to $\phi(\tau)$, such that whenever $\phi(\tau)$ exceeded this threshold, a word would be assumed to be recognized. Obviously, the lower the threshold, the higher the probability of recognition of a valid word. Unforunately, however, a decrease in the threshold also increases the probability of false alarm, *i.e.*, the recognition of a word when one is not present. The best threshold setting would be the optimum compromise between these two probabilities.

II. Binary Correlation with a Shift Register

A binary shift register may be used to approximate binary correlation. In essence, the register itself functions as the template, while a digital version of the signal continuously shifts through the register. The digitizing of the binary signal is accomplished by continuously dividing the signal into consecutive segments, or digitizing periods, of length T_q, with the weight of the mth period being

$$W_m = \frac{1}{T_q}\int_{(m-1)\,T_q}^{mT_q} f(t)dt.$$

The weight of any given digitizing period can vary from a minimum of zero, corresponding to steady space, to a maximum of one, corresponding to steady mark. A weight threshold W_T is chosen, such that if $W_m \geq W_T$, the mth digitizing period is assigned a one or mark, and if $W_m < W_T$, the mth digitizing period is considered to be a zero or space.

The digitized input signal is then shifted down an

n-bit shift register, the shift frequency being $1/T_q$ cycles per second. Since the register acts as the template, it must be T seconds in length. Thus

$$n = T/T_q;$$

i.e., the number of bits in the shift register equals the number of digitizing periods in the word to be recognized. Hence the word also may be digitized into n periods; moreover, each digitizing period corresponds to a bit in the shift register. In particular, the first digitizing period corresponds to the output bit, or nth bit of the register; the second period to the $(n-1)$th bit; etc.

The correlation process occurs between shifts of the register. The product $f(t+\tau)g(t)$ consists of comparing the state of every bit with the permanent state of the corresponding digitizing period of $g(t)$. If agreement exists, the product is one; disagreement yields a product of zero. The integral over the word length reduces to the sum of these n products. The value of the correlation function thus obtained may vary between zero and n, the latter occurring when total agreement is obtained.

The shift register always contains a digitized version of the preceding T seconds of $f(t)$. When the presence of a word in the shift register is detected by the correlation function having exceeded its preassigned threshold, the word may be read out of the register in a serial or a parallel fashion, or in any combination thereof. The information content of the word is then extracted from the digitized version of the word. Since the information is dependent on the smallest pulse width in the word, the digitizing period must be small enough such that the presence or absence of that pulse may be detected in the digitized word. This sets an upper bound on T_q; viz., it must be substantially less than the smallest information pulse.

It is apparent that the shift register technique is only an approximation to the integration of correlation, similar to the familiar trapezoidal and Simpson approximations. The probable error of the approximation decreases as the digitizing period decreases. There is, however, a realistic lower bound to T_q, the reciprocal of the bandwidth of $f(t)$. In other words, T_q must be greater than the transition times of $f(t)$ between binary states. Once this bound is reached, no further reduction in the probable error may be attained by decreasing T_q.

Summarizing, T_q is bounded both from above and below, while the probable error of the correlation approximation varies directly with T_q. Since $n=T/T_q$, it follows that the number of bits in the shift register varies inversely with the probable error of the correlation function. Often the attainment of a tolerable probable error demands an extremely large number of bits in the shift register. The utilization of bistable elements, such as vacuum tube or transistor flip-flops or magnetic cores, as the individual bits of the register may result in a prohibitively large cost or size.

This paper describes a method of simulating the correlation shift register by means of an analog delay element and a small amount of control circuitry. The delay-line shift register affords the advantage of a low correlation error without the disadvantages of cost and size inherent in a long, bistable element register.

III. The Theory of the Delay-Line Shift Register

The crux of the delay-line shift register is the continuous circulation of a time-compressed version of the preceding T seconds of the digitized signal. Although many ratios of delay to digitizing period could accomplish this end result, one ratio in particular will be discussed first, after which the effect of other ratios will be considered.

Let us consider a delay of $(n-1)T_q/n$. After each digitizing period of $f(t)$, a pulse of positive or negative polarity is launched down the delay line, depending on whether the digitizing period was a mark or a space, respectively. At present the width of the pulse is immaterial, provided it does not exceed T_q/n seconds. The pulse corresponding to the first digitizing period of $f(t)$ is launched down the line at $t=T_q$; the pulse for the second period at $t=2T_q$; etc. These pulses arrive at the output of the delay line $(n-1)T_q/n$ seconds after their launching, at which time they are reapplied to the input of the line. Thus the first pulse, which was launched at $t=T_q$, begins its second pass through the line at $t=(2-1/n)T_q$, its third pass at $t=(3-2/n)T_q$, etc.; the second pulse begins its launch at $t=2T_q$, its second pass at $t=(3-1/n)T_q$, etc.; and so forth. Note that the first pulse begins its second pass T_q/n seconds before the initial launching of the second pulse, which means that the first pulse will be T_q/n seconds, or $1/(n-1)$ of the total delay, down the line when the second pulse is launched. Thereafter the two pulses will always be T_q/n seconds apart on successive passes through the line. In general, the mth pulse will be T_q/n seconds down the line on its second pass when the $(m+1)$th pulse is launched; thereafter at any point in the closed loop the mth pulse will lead the $(m+1)$th pulse by T_q/n seconds.

Returning to the first pulse, it attempts to start its $(n+1)$th pass through the line at $t=nT_q$. At this same time, however, the nth pulse is being launched down the line. The initial launching of the nth pulse overrides the $(n+1)$th launching of the first pulse, and the latter is removed from the circulating stream permanently. In general, the mth pulse makes n passes through the line, after which its place is taken by the initial launching of the $(m+n-1)$th pulse. After $t=nT_q=T$ seconds the delay line is completely filled, each initial launching of a pulse resulting in an erasure of an older pulse.

In summary, at any point in the circulatory loop the mth pulse will be followed by the $(m+1)$th pulse T_q/n seconds later. The sole exception is when the mth pulse is making its first pass through the line; it is followed by the $(m-n+2)$th pulse, which is making its

last pass through the line. The train of n pulses from $m-n+2$ to $m+1$ represents the digitized version of $f(t)$ from $t=(m-n+1)T_q$ to $t=(m+1)T_q$, or a total of $nT_q=T$ seconds of $f(t)$. Since the pulses are T_q/n seconds apart on the loop, the entire train passes in T_q seconds, resulting in a virtual time compression of $f(t)$ by a factor of n. In other words, the previous T seconds of digitized $f(t)$ circulate around the loop every T_q seconds. The n pulses beginning with the oldest pulse in the train and ending with the newest pulse may be correlated with the compressed word $g(t)$ in a serial fashion every T_q seconds. Thus the delay-line shift register simulates the actual shift register just discussed.

Many other ratios of delay to digitizing period will accomplish this same result. For example, a delay of $(n-1)T_q/[1+k(n-1)]$ allows the mth pulse to make k complete passes before the $(m+1)$th pulse is launched. The advantage of a shorter delay may be cancelled by the disadvantage of a shorter spacing of $T_q/[1+k(n-1)]$ seconds between pulses on the line. A delay of $(n-1)T_q/[k(n-1)-1]$ results in a time inversion in the time-compressed $f(t)$. Other ratios of delay to digitizing period can result in various transformations of the time-compressed signal. Although a particular transformation may prove advantageous in correlating a specific word pattern, the infinite variety of transformations will not be pursued further.

The remainder of this paper will describe the design and operation of a particular delay-line shift register, excluding the correlation circuitry. For an example, we shall design the shift register to have a word length of

$$T = 300 \pm 30 \text{ msec},$$

and a digitizing period of

$$T_q \leq 1.5 \text{ msec}.$$

To fulfill these specifications with a conventional shift register would require a minimum of

$$\underline{n} = \underline{T}/\overline{T}_q = 270/1.5 = 180$$

bistable elements. (The bar above the quantity denotes the maximum value of the quantity, while a bar below denotes the minimum value. This notation will be used quite frequently throughout this article.)

IV. Preliminary Selection of the Clock Frequency and the Delay Line

Unless adequate timing controls are designed into the delay-line shift register, acute timing problems can result. For instance, suppose the delay of the line $T_d=(n-1)T_q/n$ changes from its assigned value by an amount δT_d. After n passes through the line, each pulse will then be $n\delta T_d$ seconds out of time with the new pulse which is to take its place. A simple method of alleviating this problem is to synchronize the pulses out of the line with a train of clock pulses, thereby having to correct only for a difference of δT_d seconds. Since the separation between pulses on the line is T_q/n seconds, the frequency of the clock would be n/T_q cps. In addition, a clock of frequency $1/T_q$ cps is required to determine the digitizing periods of $f(t)$. Moreover, this slower clock must maintain complete synchronism with the faster clock, so that the insertion of a new pulse coincides exactly with the removal of the oldest pulse.

The obvious solution to this problem is to provide a master clock with a frequency of n/T_q cps. The lower-frequency clock is then obtained by dividing the master clock frequency by a factor of n, thereby assuring complete synchronism. Although frequency division by any integer n is not difficult, if the integer were an integral power of two, only $\log_2 n$ binary counters connected in cascade would be required for the division. The specifications of the example call for $\underline{n}=180$. With the choice of $n=2^8=256$, the probable error of the correlation function will probably be lower than that at $\underline{n}$, at the worst showing no change at all.

With n fixed, the minimum frequency of the master clock becomes

$$\underline{f} = n^2/\overline{T} = 2^{16}/330 = 198.6 \text{ kc},$$

while the upper limit is

$$\bar{f} = n^2/\underline{T} = 2^{16}/270 = 242.7 \text{ kc}.$$

A reasonable standard frequency within these limits is $f=200$ kc.

The choice of n and f fix the storage length at

$$T = n^2/f = 2^{16}/200 = 327.7 \text{ msec},$$

the digitizing period at

$$T_q = n/f = 2^8/200 = 1.28 \text{ msec},$$

and the delay of the line at

$$T_d = (n-1)/f = 255/200 = 1.275 \text{ msec}.$$

Since $n=256$, this delay-line shift register will replace a conventional shift register with 256 bistable elements. In this example, generous tolerances on T and T_q allow the selection of convenient values for n and f. If T and T_q had been firmly fixed, the result would have been only a less convenient frequency division factor and a peculiar, nonstandard, clock frequency. In general, determining any two of T, T_q, T_d, f, and n uniquely fixes the remaining three.

The separation between pulses on the line is 5 μsec, which is the period of the master clock, and the delay of the line is 1275 μsec. To discern the presence of a single pulse on the line, the ratio of delay to rise time for the line must be much greater than $1275/5=255$. This restriction eliminates both distributed and lumped parameter electrical delay lines from consideration. With the present state of the art, the upper limit of the delay to rise time ratio for electrical lines is in the neighborhood of 200. It follows that the line must be of

the acoustical or ultrasonic variety, among which the quartz, mercury, and magnetostrictive delay lines are prominent. Consideration of cost, size, and temperature coefficient of delay make the magnetostrictive delay line extremely well suited for application in the delay-line shift register.

V. The Magnetostrictive Delay Line

The heart of the delay-line shift register is the magnetostrictive delay line. The operation of a magnetostrictive delay line employs three principles. The first of these is the Joule effect, in which a magnetostrictive material experiences a change in physical dimensions, *i.e.*, and expansion or contraction, with a change in magnetic flux through the material. The second principle is the propagation of a mechanical strain through the material at sonic velocity. The third principle is the Villari effect, which pertains to a change in the permeability of a magnetostrictive material in the presence of a strain.

A practical magnetostrictive delay line consists of a length of nickel-alloy wire, the sonic medium, with a coil wound around each end. A magnetic field is impressed in the nickel wire at the output coil, usually by means of a permanent magnet. A change in current in the input coil will create a strain in the nickel wire under the coil, which in turn is transmitted down the line to the output coil at the velocity of sound. Upon arrival at the output coil, the strain causes a change in the permeability of the line, which in turn causes a change in the magnetic flux density under the output coil. The voltage induced in the output coil is proportional to the time rate of change of the flux-linkages of the output coil.

Fig. 2 illustrates the output voltage waveforms for several input current pulses. Of course, the output waveforms are delayed by T_d, the time delay of the line. If a long pulse of current is applied to the input [Fig. 2(a)], the output voltage consists of two distinct doublets, corresponding to the beginning and the end of the input pulse. The structure of the output voltage doublet, which may be considered as an approximate second derivative of the input, is primarily dependent on the length of the input and output coils and the distribution of the magnetic fields of the coils. As the input pulse length is decreased [Fig. 2(b) and (c)], the doublets approach one another and the interior peaks begin to coalesce. Finally [Fig. 2(d)], that input pulse length is reached where the two interior peaks merge completely, resulting in a peak of roughly twice the magnitude. This is the pulse length for which the line is designed; decreasing the input pulse length any farther will only decrease the magnitude of the center peak.

For ease of explanation in previous sections, it was stated that a positive pulse on the delay line represented a marking digitizing period, whereas a negative pulse represented a space, and that the period of the pulses on the line was 5 μsec. In the actual implementation of the theory a digitizing period of mark will be represented by

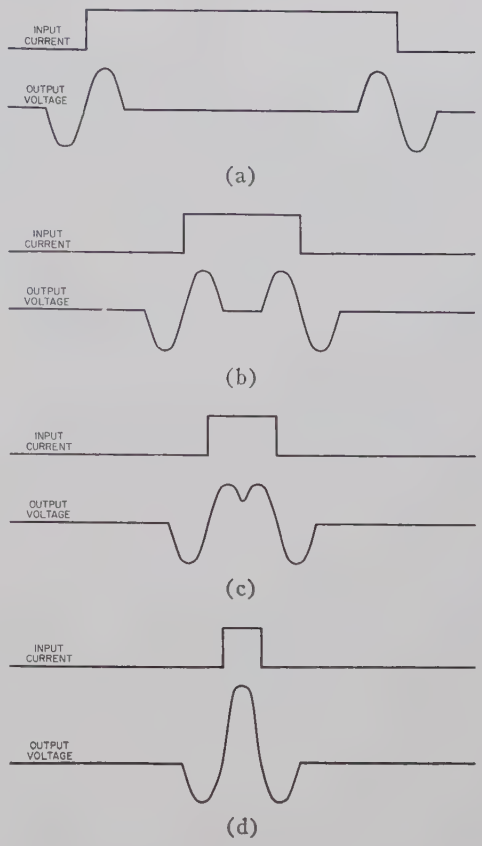


Fig. 2—Pulse response of a magnetostrictive delay line.

a 2.5-μsec pulse in the 5-μsec period, and a digitizing period of space by the absence of a pulse in the 5-μsec period. Therefore, if the delay line is designed for 2.5-μsec pulses, the condition of steady mark on the line would reveal a 200-kc square wave of current at the input and a 200-kc quasi-sinusoidal voltage at the output. The actual magnetostrictive delay we shall employ is 1274.375 μsec, not the previously calculated 1275 μsec. In addition, the master clock frequency will be doubled from 200 kc to 400 kc. These two changes will be explained subsequently.

VI. Logical Schematic of the Delay-Line Shift Register

Fig. 3 is the logical schematic of the delay-line shift register, and Figs. 4 and 5 are typical timing diagrams for the register. These figures assume that the more positive level is logical one, and that flip-flops can only be switched by a leading edge, *i.e.*, a transition from zero to one. In Fig. 3, the 400-kc master clock, the ten flip-flops A through J, AND gates 1 and 4, and the inverter provide all of the timing waveforms for the register. The set and reset outputs of the master clock are opposite polarities of a 400-kc square wave. The flip-flops A through I are cascaded binary counters, performing binary frequency division of the 400-kc master clock waveform. Table I lists the frequency and period of either output of the important stages in the chain.

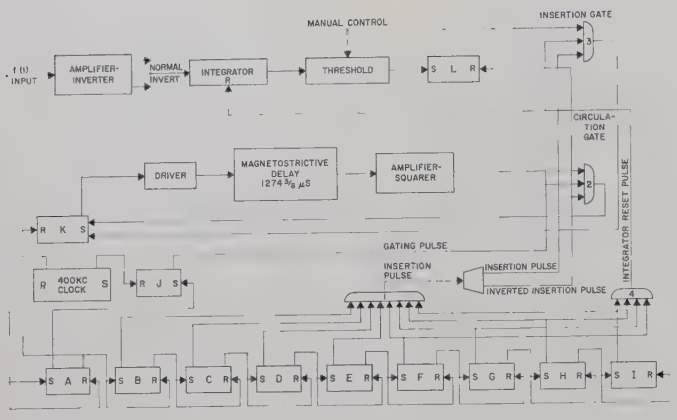


Fig. 3—Logical schematic of the delay-line shift register.

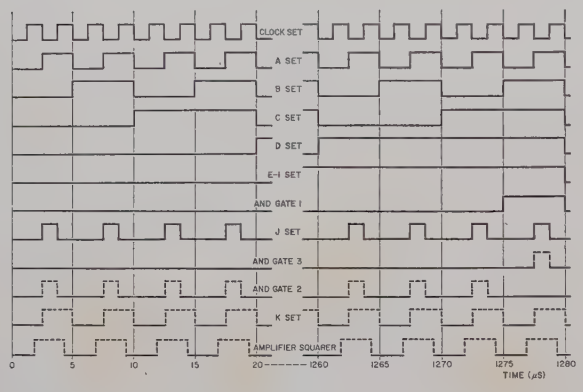


Fig. 4—Typical waveforms for a digitizing period.

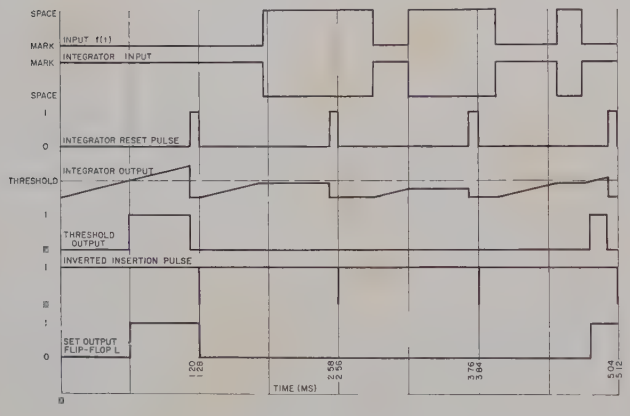


Fig. 5—Timing diagram of the digitizing section.

TABLE I

Stage	Frequency (cps)	Period (μs)
Clock	400,000	2.5
A	200,000	5
E	12,500	80
I	781.25	1280

Flip-flop *J* provides the gating pulses which synchronize the output of the delay line amplifier-squarer. Flip-flop *J* is set by the set output of *A* and reset by the set output of the clock (see Fig. 4). Flip-flop *J* then receives a leading edge on its set input every 5 μsec and two leading edges on its reset input between set inputs. The leading edge on the set input always sets *J*. The first reset input, which occurs 1.25 μsec after the set of *J*, resets *J*, and the second reset input, occurring 2.5 μsec after the reset of *J*, has no effect since *J* has already been reset. The resultant set output of flip-flop *J* is a 1.25-μsec pulse with a period of 5 μsec, the leading edge coinciding with the leading edge of the set output of *A*. The set output of *J* is called the gating pulse.

AND gate 1, which gates the set outputs of flip-flops *B* through *I*, furnishes the insertion pulse, and the inverter which follows provides both the insertion pulse and the inverted insertion pulse. The insertion pulse is a 5-μsec pulse which occurs every 1.28 msec, and its trailing edge separates the successive digitizing periods of $f(t)$. The output of AND gate 4 is the integrator reset pulse, an 80-μsec pulse occurring every 1.28 msec. The trailing edge of the integrator reset pulse coincides with that of the insertion pulse, and hence with the separation between digitizing periods.

Fig. 5 shows the timing of the digitizing section for a typical binary signal input and a particular threshold setting. The digitizing section is composed of the amplifier-inverter, the integrator, the threshold, and flip-flop *L*. The input to the digitizing section is the binary signal $f(t)$. The amplifier-inverter standardizes the binary levels of the signal, providing both polarities of $f(t)$ as outputs. With the normal-invert switch, one may select either polarity of $f(t)$ as the mark level. The integrator determines the weight of each digitizing period by integrating the signal over the period. Since a finite time is required to reset the integrator for the next digitizing period, the integration actually occurs only during the first 1.20 msec of the digitizing period, and the reset during the last 0.08 msec of the period. The integrator reset pulse accomplishes this periodic reset. Provision is shown for manually setting the weight threshold to a desired value. If the output of the integrator exceeds the preset threshold, the threshold output changes from zero to one, setting flip-flop *L*. Flip-flop *L* is periodically reset at the end of the digitizing period by the inverted insertion pluse. Thus, flip-flop *L* is set only during the digitizing periods which have been designated as marking periods by the weight integral having exceeded the preassigned weight threshold. The flip-flop is set whenever the instantaneous value of the integral exceeds the threshold and is reset at the termination of the 1.28 msec digitizing period.

AND gate 2, the circulation gate, and AND gate 3, the insertion gate, control the insertion of the new pulse into the circulating stream, the removal of the oldest pulse in the stream, and the gating of the pulses out of the delay line amplifier-squarer. Since one of the inputs

to the insertion gate is the insertion pulse and one of the inputs to the circulation gate is the inverted insertion pulse, no pulses can pass through the circulation gate during the insertion pulse, *i.e.*, the last 5 μsec of every digitizing period, while this is the only time a pulse may pass through the insertion gate. Since the oldest pulse in the train arrives at the circulation gate during the insertion pulse, it is denied passage through the gate and thus erased from the train. Simultaneously, a gating pulse arrives at the insertion gate during this 5-μsec interval. If the digitizing period is a mark, the remaining input to the insertion gate will be enabled, allowing the gating pulse to pass through the insertion gate and subsequently into the circulatory stream. A digitizing period of space inhibits the passage of the gating pulse through the gate. In this manner, the new pulse, or lack of pulse, is slipped into the stream while the oldest pulse, or lack of pulse, is erased.

With a clock frequency of exactly 400 kc and a delay of precisely 1274.375 μsec, the 1.25-μsec gating pulses will occur precisely in the center of the 2.5-μsec pulses out of the amplifier-squarer. Hence, if the delay is off relative to the clock frequency, the gating pulses will synchronize the delayed pulses through the circulation gate. With a constant clock frequency, the delay of the line may vary ± 0.635 μsec from the nominal value, or approximately 5 parts in 10^4, without the gating pulses slipping out of the delayed pulses from the amplifier-squarer.

The pulses from the circulation gate and the insertion gate are only 1.25 μsec wide, the width of the gating pulses. Flip-flop K effects the merger of the newest pulse into the circulatory stream, as well as the stretching of the pulses to their delay line width of 2.5 μsec. Flip-flop K is periodically reset 1.25 μsec after the trailing edge of each gating pulse by the reset of A, while it is set by the output of either AND gate 2 or 3. The merger is then accomplished by the dual set input, and the pulse stretching by the delayed reset.

The leading edge of each pulse into the line coincides with the leading edge of a gating pulse. Since the delay of the line is 1274.375 μsec and the period of the gating pulses is 5 μsec, each pulse into the line is delayed by $1274.375/5 = 254.875$ periods. This fractional period delay causes the delayed pulse to arrive at the output slightly ahead of a gating pulse, the lead time being $0.125(5) = 0.625$ μsec. Thus the 1.25-μsec gating pulse must occur precisely in the center of the 2.5-μsec output pulse.

VII. Conclusions

The usage of a closed-loop delay line as a storage element can hardly be considered a new idea. It has been employed in many fields of electronic endeavor—for example, in digital computers—and is generally viewed as a fundamental technique of storage. However, the variation of this basic theme to the task of correlating a binary signal with a known word via the route of time compression must be considered novel.

The evolution from the original idea to the operative result necessarily required that many problems of a practical nature be solved. Among these was the problem of determining during each digitizing period whether the binary signal was a mark or a space. A very simple solution to this problem, and one that is frequently used in equipments of a similar nature, is to base the determination on a single sample or strobe of the input signal during the digitizing period. The accuracy of this method obviously cannot compare with that afforded by the weight integral. The reset integrator and the threshold of the digitizing section represent the practical embodiment of the theory of the weight integral.

Another problem which arose concerned the task of synchronizing the digitizing period with the period of the pulses on the line, such that the oldest pulse position in the train arrived at the output of the line and the digitizing period came to a close simultaneously. The use of a single, master clock with cascaded binary counters and appropriate gates to determine both the line pulse period and the digitizing period alleviated this problem, but presented still another, viz., maintaining this synchronism over expected variations in the clock frequency and the delay of the magnetostrictive line. This latter problem was solved by strobing the pulses out of the line with smaller gating pulses centered in the larger pulses.

Still another major problem was the amplification and squaring of the low-level pulses out of the delay line in the presence of the background noise of the equipment. The use of a differential amplifier in the amplifier-squarer cleared up this difficulty.

In summary, the author feels that the technical significance of this effort lies in the unique application of an established technique and in surmounting the inevitable problems which impede the evolution of an idea to an actuality.

As to the performance of the delay-line shift register, all expectations were fulfilled. A striking example of the storage facet of the register was obtained by displaying one complete digitizing period of the circulation gate output on an oscilloscope, initiating the sweep at the beginning of each successive digitizing period. Each sweep of the oscilloscope then displayed the time-compressed version of the preceding word length of the binary signal. The effect was that of viewing the contents of a slot exactly one word length in width, which was being moved across a plot of the input signal in the direction of increasing time. Consequently the input signal paraded through the slot from right to left.

Finally, the delay-line shift register did prove to be cheaper and oft times to occupy less space than shift registers utilizing conventional bistable devices. To construct, for example, a 256-bit shift register employing junction transistor flip-flops would require 512 transistors for the register alone, whereas the delay-line shift register used only 47 transistors. This savings in transis-

tors alone could pay for the magnetostrictive delay line many times over, not to mention the space consumed by 256 flip-flops. If magnetic cores are used in the register, the size of the register could perhaps be made comparable to that of the delay-line shift register, but the cost of the 256 cores and the associated driving circuitry would still be prohibitive. In addition, while the cost of the conventional register is roughly proportional to the number of bits, the cost of the delay-line shift register is relatively independent of the number of bits when this number is large. For this reason, the delay-line shift register certainly offers a cost savings over long conventional registers.

Selected Bibliography

[1] G. I. Cohn, *et al.*, "Magnetostrictive delay line for video signals," IRE Trans. on Component Parts, vol. CP-5, p. 53–59; March, 1958.
[2] H. Epstein and O. B. Stram, "A high-performance magnetostriction delay line," IRE Trans, on Ultrasonics Engineering, vol. UE-5, pp. 1–24; August, 1957.
[3] S. Morleigh, "A survey of delay lines for digital pattern storage," *Electronic Engrg.*, vol. 30, pp. 380–387; June, 1958.
[4] A. Rothbart and L. Rosenberg, "A theory of pulse transmission along a magnetostrictive delay line," IRE Trans. on Ultrasonics Engineering, vol. UE-6, pp. 17–31; December, 1957.
[5] T. B. Thompson and J. A. M. Lyon, "Analysis and application of magnetostriction delay lines," IRE Trans. on Ultrasonics Engineering, vol. UE-4, pp. 8–22; August, 1956.
[6] R. C. Williams, "Theory of magnetostrictive delay lines for pulse and continuous wave transmission," IRE Trans. on Ultrasonics Engineering, vol. UE-7, pp. 16–38; February, 1959.

Proposal for Magnetic Domain-Wall Storage and Logic*

DONALD O. SMITH†

Summary—**Magnetic storage of binary digital information is currently accomplished by utilizing the two stable and degenerate energy states corresponding to opposite spatial directions of a magnetic domain. However, a second type of magnetic-energy degeneracy exists since a domain wall separating two domains has the same energy regardless of the sense (clockwise or counter-clockwise) of rotation of the magnetization M within the wall. Hence, the rotational sense of M in passing through a wall can be used to represent binary information. The two methods of representation are essentially different since different aspects of spatial symmetry are involved. A practical method of utilizing wall-rotation information coding can be effected by means of strips of thin magnetic film and nearby current conductors which are used to generate magnetic fields in specified directions at specified times and places. A shift register consists of a regular up-and-down domain pattern with M perpendicular to the long strip axis; no information is contained in the domain pattern itself, which only serves to separate spatially the information-bearing walls. Shifting is accomplished by two-phase clock pulses on a pair of shifting windings. Read-out requires the use of the operation of conditional erase, accomplished as follows: if two walls of the same sense are brought together they form a double wall which can subsequently be separated into two walls; if the walls are of the opposite sense they will destroy one another upon being brought together. Thus, erasure occurs on condition that two walls have opposite sense. Read-out then consists of conditional erase between a data wall and a wall of known sense read-in for the purpose of read-out. By introducing junctions between strips, fan-out can be achieved by moving a wall from a single to a double strip. Furthermore, if data walls from two independent converging strips are juxtaposed in a third strip, the logical operations AND or OR can be performed by appropriate use of the operations of conditional erase and the additional operation of unconditional erase; unconditional erase erases two walls regardless of sense. Finally, a method of complementing information can be conceived by considering the details of flux closure within a domain wall.**

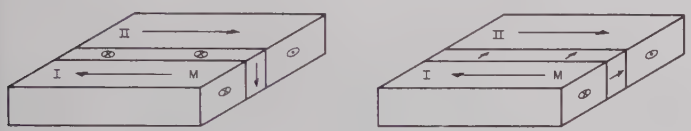
* Received by the PGEC, July 5, 1961.
† Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass. Operated with support from the U. S. Army, Navy and Air Force.

Introduction

MAGNETIC storage as currently accomplished makes use of the fact that if a magnetic domain configuration is stable, so is the one which corresponds to a reversal in direction of all those domains. This utilization of two antiparallel directions in space of equivalent magnetic stability is common to all of the present forms of magnetic storage such as tapes, drums, disks, cores or thin films. However, an essentially different kind of magnetic representation of binary information is possible. The essential point is that for any domain configuration the magnetic stability is independent of the sense of rotation of the magnetization within the domain walls which separate the domains; hence, the two possible senses of rotation can be used to represent binary information. Furthermore, logical operations can be performed by using the fact that if two walls of opposite sense are brought together they will destroy one another but if of the same sense they will form a double wall which can be separated into two walls again.

Principle of Domain-Wall Storage

The concept of a domain wall in a ferromagnetic material is illustrated in Fig. 1. Domains I and II are uniformly magnetized and are separated by a domain wall in which the magnetization is rapidly varying in direction. Two limiting cases of the spatial variation of the magnetization $\boldsymbol{M}$ within the wall are shown in Fig. 1(a) and 1(b). In Fig. 1(a) $\boldsymbol{M}$ rotates in the plane of the wall (Bloch wall[1]); in Fig. 1(b) the rotation is perpendicular to the wall (Néel wall[2]). In either case the rotation from domain I to II can be clockwise or counter-clockwise, and it is this sense of rotation which is being considered here as a medium for information storage. Clockwise or counter-clockwise walls will be referred to as positive P or negative N, respectively.



(a) (b)

Fig. 1—Domain walls in a ferromagnetic material. (a) Bloch wall. (b) Néel wall.

Whether or not walls are of the Bloch or Néel (or other) type depends essentially on geometry. For example, for films of thicknesses in the region near 200 A or 2000 A, the walls are Néel or Bloch, respectively;[3] for intermediate thicknesses, walls in films may be more complicated, as is the case for the well-known "cross-tie" wall.[4] The following discussion of storage and logic is immediately applicable to either Néel or Bloch walls. Use of other types of walls is not precluded, but has not been considered in detail.

Basic Operations

The simplest utilization of domain-wall coding would be in a shift register; thus methods of read-in, shifting and read-out are required. The operation AND and OR are also conceptually possible so that in principle a complete computing scheme is possible. Finally, a method for complementing will be described.

Read-in and Shifting

In order to be useful as a storage medium, walls of known sense must be injected into a magnetic material. A method of doing this is proposed in Fig. 2. Fig. 2(a) is a top view of a strip of magnetic film magnetized in the plane of the strip and perpendicular to the long axis of the strip; a magnetic field $\boldsymbol{H}$ is applied locally to reverse the magnetization $\boldsymbol{M}$ between the dotted lines. After local reversal two Néel walls of opposite sense will have been generated as shown in Fig. 2(b). If $\boldsymbol{H}$ had been applied in a direction to cause rotation in the opposite direction [dotted vector in Fig. 2(a)], the injected walls would have opposite senses, *i.e.*, the N-wall would be a P-wall and vice versa.

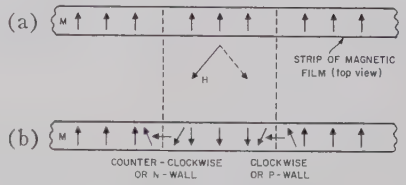


Fig. 2—Injection of Néel walls of known sense into a magnetic film. (a) Before injection. (b) After injection.

If the magnetic strip is thick enough to support Bloch walls, it can still be expected that walls of definite and opposite sense will be created. This is so because during the dynamic process of wall formation the torque on the magnetization due to $\boldsymbol{H}$ is $\boldsymbol{M}\times\boldsymbol{H}$ and is in a direction perpendicular to the plane of the film, *i.e.*, parallel to the plane of a Bloch wall; since $\boldsymbol{M}\times\boldsymbol{H}$ is of opposite sign for the solid or dotted vectors of Fig. 2(a), the senses of the injected Bloch walls should reverse for the two cases.

A method of shifting a number of bits along a strip is illustrated in Fig. 3. A long magnetic strip is magnetized alternately in an up-and-down pattern of closure domains. No information is contained in this regular pattern which serves merely to space the information which is stored in the sense of rotation of the walls (either Bloch or Néel) separating these domains; a pattern of information *PNNPP* is shown in Fig. 3(a). The proposed domain configuration may be stable enough for operation without further elaboration; however, if necessary, a magnetic easy axis perpendicular to the long axis of the strip can be introduced to provide further stabilization. In fact, it may prove difficult to make a film without an easy axis, in which case the proposed orientation will be the most favorable one. In order to shift the information to the right, local magnetic fields $\boldsymbol{H}$ in the vicinity of the walls and having the directions shown in Fig. 3(a) must be provided; a possible concomitant magnitude variation of $\boldsymbol{H}$ between walls is shown in Fig. 3(b). Application of the shifting fields will move the walls to the right to the stable positions of zero $\boldsymbol{H}$ as shown in Fig. 3(c).

Physical realization of the scheme illustrated in Fig. 3 is difficult since no simple way is available to supply the local field pattern required for shifting. A more prac-

[1] F. Bloch, "Zur Theorie des Auslauschproblems und der Remenenzerscheinung der Ferromagnetika," *Z. Physik*, vol. 74, pp. 295–335; February, 1932.

[2] L. Néel, "Remarques sur la theorié des Propriétés Magnétiques des Couches Minces et des grains fins," *J. Phys. Radium*, vol. 17, pp. 250–255; March, 1956.

[3] S. Methfessel, S. Middlehoek, and H. Thomas, "Domain walls in thin Ni-Fe films," *IBM J. Res. and Dev.*, vol. 4, pp. 96–106; April, 1960.

[4] E. E. Huber, Jr., D. O. Smith, and J. B. Goodenough, "Domain-wall structure in permalloy films," *J. Appl. Phys.*, vol. 29, pp. 294–295; March, 1958.

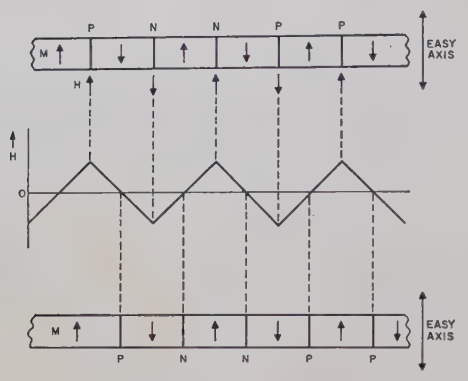


Fig. 3—Shifting to the right. (a) Shifting to the right. (b) Local fields for shift right. (c) Shifted *PNNPP*.

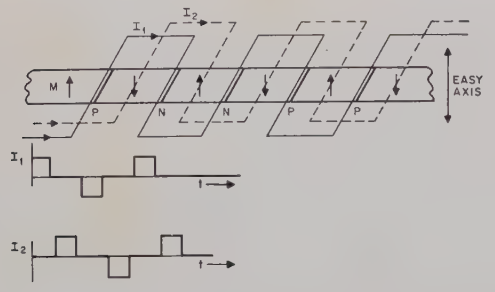


Fig. 4—Practical configuration for injection and shifting.

tical scheme is shown in Fig. 4 which shows the top view of a ferromagnetic strip and two windings placed in nearby planes parallel to the film plane; the walls are now at an angle to the easy axis which is perpendicular to the magnetic strip. A pulse of current in winding I will provide a local field ***H*** perpendicular to the walls with a vertical component which will move the walls to the right and position them under winding II; a subsequent current pulse in winding II will move the walls to the right into position under winding I, etc. . . .

Conditional and Unconditional Erase

The operation of conditional erase takes advantage of the fact that two walls of opposite sense if moved together will destroy each other; however, walls of the same sense when brought together do not necessarily destroy each other, but can form a double wall which can subsequently be separated again into two single walls. Thus erasure occurs on condition that the walls have opposite sense. An illustration of conditional erase is shown in Fig. 5. In Fig. 5(a) two similar walls (either *PP* or *NN*) are brought together and subsequently separated by the application of suitable local fields; Fig. 5(b) shows the case of two dissimilar walls (either *PN* or *NP*) which destroy one another when brought together. In the practical realization the slanted wiring and domain configuration of Fig. 4 would be used.

Unconditional erase requires that two adjacent walls be removed regardless of sense. Thus a field must be applied such that the pattern of Fig. 5(a)-Ab is changed into that of Fig. 5(b)-Bc. It is clear that considerably

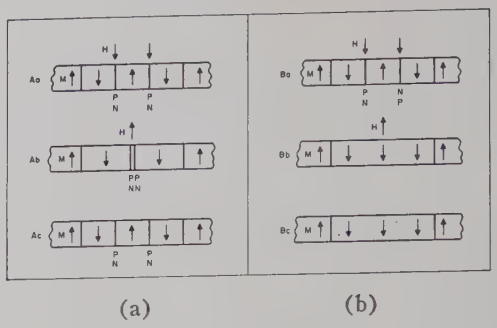


Fig. 5—The operation of conditional erase. (a) Interaction of *PP* or *NN* walls. (b) Interaction of *PN* or *NP* walls.

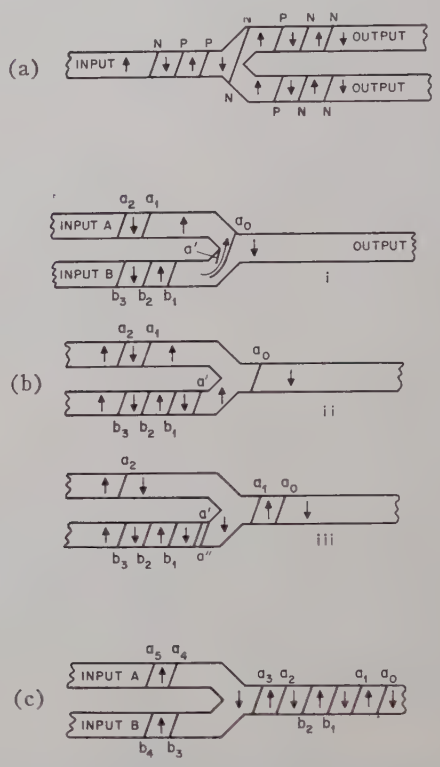


Fig. 6—Fan-out and juxtaposition. (a) Fan-out. (b) Junction behavior. (c) Juxtaposed data.

larger fields will be required for unconditional as compared to conditional erase, thus providing for an operating margin in any practical system.

Fan-Out and Juxtaposition

Fan-out is shown in Fig. 6(a). Information in an arbitrary pattern of walls is shifted from one strip into two parallel strips at a junction as shown; the energy required to create the second wall comes from the current in the shifting windings. There do not appear to be any conceptual problems associated with this operation.

Juxtaposing data from two input strips A and B into an output strip is more difficult [Fig. 6(b)]. Since a regular sequence of up-and-down carrier domains must be preserved, one finds that it is necessary to transfer by pairs of bits from A or B; the dummy bit a_0 must

also be added to keep the carrier sequence correct. In Fig. 6(b)-i, the a_0 wall is shown in the process of moving into the output strip; note the postulated appearance of the wall a' which appears in order to separate the "up" domain on the right of wall a_1 from the "down" domain on the right of wall b_1. In Fig. 6(b)-ii, the a_0 wall is in the output strip and the a' wall is temporarily placed in the B strip. After the transfer of a_1 from A, a_1 and a_0 are in the output and two auxiliary walls a' and a'' are in the B strip [Fig. 6(b)-iii]. It appears that a' and a'' will be of the same sense regardless of the senses of a_0 and a_1, and hence that before transferring from the B strip a' and a'' must be unconditionally erased. The situation after three pairs of transfers is shown in Fig. 6(c) which shows the juxtaposition of bits a_1–b_1 and b_2–a_2.

Read-out

A detailed discussion of read-out will not be attempted here. Conversion from wall-rotation coding to domain-direction coding could be performed by the operation of conditional erase between the data walls and walls of known sense injected for this purpose. Subsequent data conversion might be by means of magneto-optical effects or voltages induced by domain switching.

A useful shift register is now conceivable by utilizing the operations of read-in, shifting, fan-out and conditional erase.

A shift register made from thin magnetic films but which uses domain-direction coding has previously been described by Broadbent.[5]

AND and OR

The logical functions AND and OR (Table I) can be performed by the sequence of operations a) juxtaposition, b) conditional erase, and c) read-in. The resulting wall patterns for all combinations of b_1 and a_1 are shown in Table II, which should be compared with Table I. Note that the read-in operation must not change the pattern P–P or N–N; this appears possible from the conception of the read-in process previously described. Note also that the choice of reading in P–N or N–P depends on which function (AND or OR) is being performed and which wall (left or right) is to represent the result. Finally, processing must take place on two pairs of bits in order to eliminate the unwanted walls in pairs [Table II (d and e)].

Complementation

Complementation might be accomplished with offset strips as shown in Fig. 7. If the correct strip dimension and film thickness are used, a Néel wall entering the wide region should become two Néel segments in order to minimize the magnetostatic energy associated with the wall analogous to the local flux closure generated in cross-tie walls.[4] The lower Néel segment is the complement of the upper, and hence the output will be the complement of the input.

TABLE I

LOGICAL AND AND OR

AND	OR
$P \cdot P = P$	$P \vee P = P$
$N \cdot N = N$	$N \vee N = N$
$P \cdot N = N$	$P \vee N = P$
$N \cdot P = N$	$N \vee P = P$

TABLE II

DOMAIN-WALL REALIZATION OF AND AND OR

Operation		Wall Pattern			
a. Juxtaposed data		P N	P N	P N	N P
		b_1	a_1	b_1	a_1
b. Conditional erase		P N	P N		
c. Read-in	AND	P N	P N	Keep right wall: P	N
				Keep left wall: N	P
	OR	P N	P N	N	P
				P	N
d. Processed data					
		$c_2 = a_2$		$c_1 = b_1$	
		AND/OR b_2		AND/OR a_1	
e. Unconditional erase of two center walls		c_2		c_1	

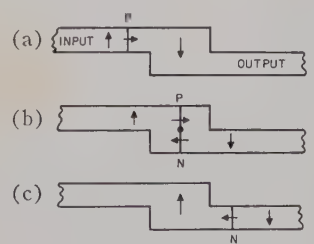


Fig. 7—Complementation.

ACKNOWLEDGMENT

It is a pleasure to acknowledge discussions with J. I. Raffel and W. A. Clark.

[5] K. D. Broadbent, "A thin magnetic film shift register," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-9, pp. 321–323; September, 1960.

Cryosar Memory Design*

R. C. JOHNSTON†, MEMBER, IRE

Summary—The compensated cryosar is a negative-resistance two-terminal device utilizing a bulk effect in germanium at liquid helium temperatures. Its bistable nature, and the ease with which it can be fabricated in large arrays recommend it for application to computer memory systems. However, careful consideration of device and circuit parameters is necessary if a successful large memory is to be achieved.

Introduction

THE cryosar[1] is a two-terminal device utilizing impact ionization of impurities in germanium at liquid helium temperatures. Impact ionization is a bulk effect, and elements may be made merely by placing ohmic contacts on opposite sides of an appropriately doped germanium wafer.

The current in this device is quite small until a critical voltage is reached. At that point the existing free carriers gain sufficient energy in the electric field to ionize the impurities upon impact.[2] The voltage then remains constant while the current increases by perhaps four or five orders of magnitude, at which time all the carriers have become ionized and the resistance increases. If uncompensated germanium is used (having only one type of impurity present) a characteristic similar to that of back-to-back Zener diodes is obtained as shown in Fig. 1.

A much wider range of applications has been opened up, however, with the discovery[1] that with compensated germanium there is a region of negative resistance between the high- and low-impedance states, permitting bistable operation. This is a current-dependent region (see Fig. 1), as opposed to the voltage-dependent region of the tunnel diode.

The prebreakdown resistance, peak voltage V_1, valley voltage, and holding current I_1 are all designable quantities, being functions of the wafer thickness, the contact area, and the impurity concentration. The voltages scale directly with the thickness, and a cryosar made twice as thick has exactly twice the voltages. The holding current does not scale with the contact area. The current flows in a filament, and the major effect of reducing the contact area is a reduction in capacitance and prebreakdown conductance.

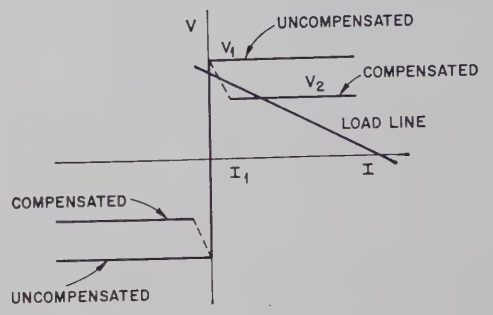


Fig. 1—*V-I* characteristic of simple cryosars with compensated and uncompensated germanium.

Basic Memory Principles

The cryosar, being a simple bistable device, was immediately considered as a computer-memory component. The ease with which it can be fabricated in large arrays makes it well-adapted to the random-access memory, which is typically a repetitive structure with simple interconnections. A load line can be placed so as to intersect the characteristic at two stable points as shown in Fig. 1, one at low current and the other at high current. The state of this bistable device can be changed by pulsing the load line above the peak voltage or below the valley voltage. For matrix operation a separate load resistance must be in series with each element. If a common load resistance were used, external to the matrix, there would be interaction between the elements sharing the load resistance. A compound version[3] of the cryosar has been developed to include these series resistors while retaining the ease of array fabrication of the simple cryosar. Fig. 2 shows the formation of the compound cryosar by sandwiching together two dissimilar cryosars. Cryosar A is a compensated type while cryosar B is uncompensated with a high prebreakdown conductivity. There are three positive-resistance regions, which are called the ZERO, ONE, and TWO (or READ) regions. Since this is a bilateral device it has identical characteristics in the third quadrant; however, these are not used. By raising the break point of cryosar B up quite high or by using a linear resistance as element B, a single-break version of the compound cryosar may be obtained which has a higher resistance in the ONE region than the simple cryosar.

The compound-cryosar characteristics are shown in

* Received by the PGEC, February 23, 1961.

† Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass. Operated with support from the U. S. Army, Navy and Air Force.

[1] A. L. McWhorter and R. H. Rediker, "The cryosar—a new low-temperature computer component," PROC. IRE, vol. 47, pp. 1207–1213; July, 1959.

[2] N. Sclar and E. Burstein, "Impact ionization of impurities in germanium," *J. Phys. Chem. Solids*, vol. 2, pp. 1–23; March, 1957.

[3] R. H. Rediker and A. L. McWhorter, "Compound cryosars for low-temperature computer memories," *Solid State Electronics*, vol. 2, pp. 100–105; March, 1961.

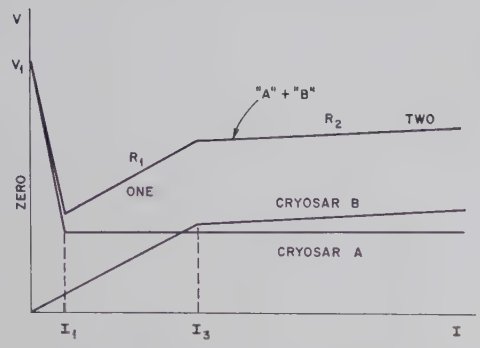


Fig. 2—The compound cryosar is the series combination of two simple cryosars.

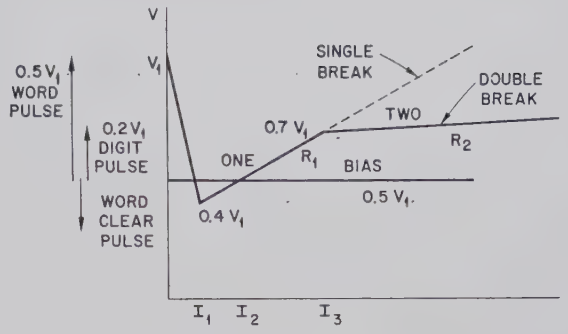


Fig. 3—Cryosar characteristic and pulse amplitudes as used in coincident-voltage, nondestructive-read memory.

Fig. 3 together with the load line and pulse amplitudes as used in a coincident-voltage, nondestructive-read memory. The numerical values shown in the figures are not fixed requirements, as both the circuit and the device designs are flexible, but represent a compromise in the factors involved. The operation of the memory may be explained with the aid of Fig. 4 which shows a cryosar matrix. The word lines are shown beneath the wafer, and the digit lines on top. A bias voltage of $0.5\,V_1$ volts is applied between the word and digit lines. Now, if a word line is pulsed with a polarity opposing the bias (negatively), any elements in that word storing a ONE will be switched to a ZERO as the load line goes below the valley point. This is called the CLEAR operation. Coincident voltages are used only for writing, which is always preceded by the CLEAR operation. A word line is pulsed positively, and the digit lines are pulsed negatively or not, depending on the information to be stored. The coincidence of the $0.5\,V_1$ word pulse and the $0.2\,V_1$ digit pulse gives a pulse of $0.7\,V_1$. Only those elements receiving both pulses are switched to the TWO state and come to rest in the ONE state upon removal of the pulse. In the READ operation an entire word is read at once by application of a positive pulse to a word line. Those elements holding a ONE are switched into the TWO state and back nondestructively to the ONE state.

The zero-resistance bias load line is necessary because any load resistance would be common to all the elements on a line, and the bias voltage would be a function of the stored information. During pulses, the load line has the slope $(R_{TS}+R_{TL})$, where R_{TS} and R_{TL} are the resistances seen from the memory element on the word and digit sides, respectively. The zero-resistance bias load line may be supplied by inductances shunting R_{TL} and R_{TS} as shown in Fig. 4. They may not be necessary in actual practice because R_{TL} and R_{TS} must be quite small for other reasons and the change in voltage with stored information may be neglected in most cases.

It is also possible to make a memory with the single-break cryosar of Fig. 3. The concepts of concident write,

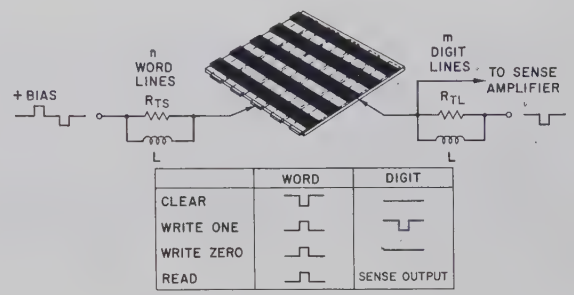


Fig. 4—An n by m cryosar matrix with word and digit lines.

and nondestructive read still apply. The digit pulse would not be clamped by the TWO state and could be increased to $0.5\,V_1$, giving decreased switching times. The relative merits of the single- and double-break cryosar memories will be discussed in detail later.

Experimental Work

The placement of ohmic stripe contacts on opposite sides of a germanium wafer has been accomplished both by evaporation and by plating. An example of this is the evaporated contact, 32 by 32 matrix shown in Fig. 5. It is encouraging to note that a 100 per cent yield was obtained. The cryosars obtained were of the simple type shown in Fig. 1; the compound type shown in Fig. 2 requires a more elaborate structure. The ideal method is to make a two-layer wafer with the low-resistivity layer obtained by epitaxial growth. The stripes can then be placed as before. It is necessary to etch the epitaxial layer away between stripes to avoid stripe-to-stripe conduction on the low-resistivity side.

Compound cryosar matrices of this type are not yet available, so the experimental work was performed on 5 by 5 matrices with the construction shown in Fig. 6. Germanium disks 40 mils in diameter were cut ultrasonically from a low resistivity wafer and alloyed to a high resistivity wafer with indium-coated kovar disks. Fig. 7 shows the V-I curves for a row of five from the 5 by 5 experimental matrix. Each curve is displaced one division to the right for clarity. The variations in voltages were found to be within the variation found in the wafer thicknesses.

The switching speed of the device is measured by observing the current waveforms when a rectangular input voltage of adjustable amplitude is applied. Fig. 8 shows the current responses produced by successive voltage drive pulses with amplitude changes of 1 db per step. The write overdrive in a practical memory is about 2 db over V_1, giving a switching time of 30 nsec. The read overdrive is about 1 db above the second-break voltage, giving a 40-nsec switching time. With the single-break cryosar, the read operation does not involve any switching and should have no delay. However, it is possible to speed up the switching times by using short pulses (say 10–20 nsec). The apparent threshold voltages are then increased, resulting in a faster switching time by a factor of about 3.

Fig. 5—An evaporated contact 32 by 32 matrix of simple cryosars.

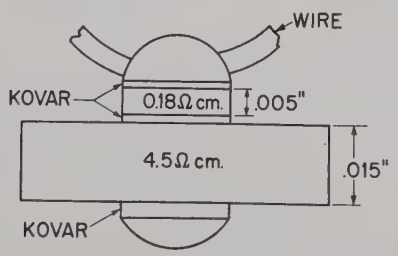


Fig. 6—Fabrication of experimental compound cryosar matrix.

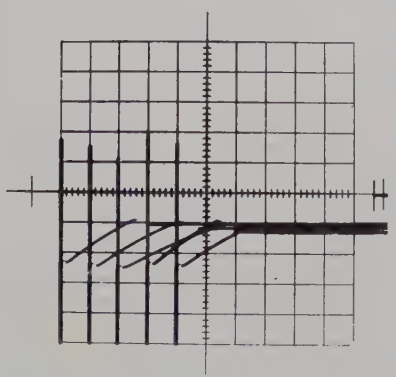


Fig. 7—*V-I* curves for five double-break compound cryosars. (Scale 0.5 v/cm vertically and 0.1 ma/cm horizontally.)

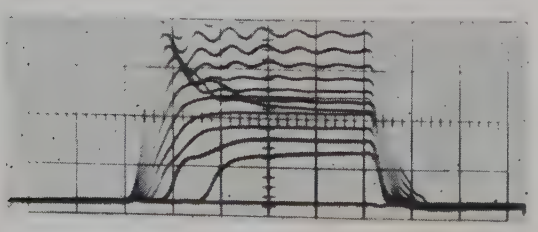


Fig. 8—Current response of compensated simple cryosar with pulse increased in 1-db steps (20 nsec/cm).

The first system experiments were made on a 16-bit memory using 15-nsec pulses and including coincident write. Operation was critical with narrow margins. It was found that longer read pulses were necessary because of the slow switching with low overdrive. Also, considerable difficulty was experienced because of the destructive reading of a ONE. The compound cryosars used had more capacitance than would be encountered in a high-density, evaporated-contact matrix, and the fall of the read pulse produced a current overshoot that switched the element back to the ZERO instead of the ONE state.

The final experiments used a more complex pulse pattern, but only on one bit and without the coincident-write operation. The pulse pattern, shown in Fig. 9, consisted of a burst of read pulses at a 10-Mc rate with 15-nsec rise and 85-nsec fall times; a clear pulse; another burst of read pulses, this time nondestructively reading a ZERO; and of a write-ONE pulse. Scales are 300 nsec/cm and 1 v/cm for input (top) and 0.1 v/cm for output (bottom).

The margin diagram for this experiment is shown in Fig. 10. The write and clear pulses were not critical, but the read pulse and bias must be in the center region for proper operation. The 2 signifies that the cryosar was in the TWO state at all times. In the *F* region the read pulse is high enough to falsely read a ZERO as a ONE. In the *N* region the read pulse is too low to switch into the TWO state. The *D* region signifies the destructive reading of a ONE. This diagram is only for one bit, and the effective diagram for a large memory would be the superposition of the diagrams for each bit. Nonuniformity in the breakdown voltages would shift the individual center regions and make the composite center region smaller.

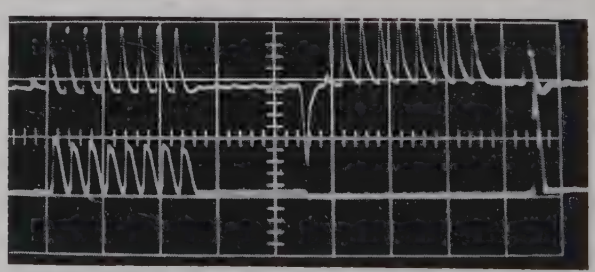


Fig. 9—Memory pulse pattern. Scales are 300 nsec and 1 v/cm for input (top) and 0.1 v/cm for output (bottom).

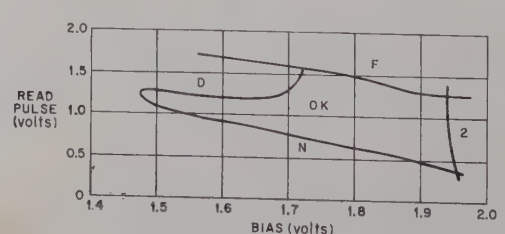


Fig. 10—Margin diagram for one bit.

Calculation of Circuit and Device Parameters

In this section the design of a large-size cryosar memory, based on the experimental background presented, will be discussed. The circuits considered will be of the external decoder type with $(m+n)$ transmission lines into the helium for an m by n matrix. A cryosar decoder circuit[4] has been described which would reduce the number of word lines to 2 $\log_2 n$. However, insufficient experimental work has been performed on it to enable design of a large decoder.

The design constraints considered first will be those based on the *V-I* characteristic, followed by those based on pulse operation, including line termination, ONE/ZERO ratio, and power dissipation. There are so many variables involved, perhaps 15 or 20, that the designs are not unique and must be based on compromises. For instance, the impedance level is set by estimating the relative difficulty of making one resistance (R_{TS}) low and another (R_1) high.

The values for the second break and valley points of $0.7V_1$ and $0.4V_1$ shown in Fig. 3 are approximately those of the experimental units, which are quite satisfactory. There are two obvious differences between the proposed characteristic (Fig. 3) and the experimental units (Fig. 7): the negative resistance at the second break, and the slope of the TWO state, R_2. A small amount of negative resistance decreases the bias margins, but increases the output signal and would be acceptable. R_2 is very low, the order of a few ohms for the experimental units. Its inclusion at the level of hundreds of ohms will be shown to be necessary from the circuit point of view. Fortunately, this does not make the fabrication more difficult. The contact areas in a high-density matrix are quite small, and the current density can be large enough to raise R_2 to the desired level.

From Fig. 3 it is seen that the maximum output signal during read approaches $0.3V_1$, since the first $0.2V_1$ volts of the $0.5V_1$ input pulse is dropped across R_1. This signal is of such polarity as to oppose the bias on the $n-1$ elements on the digit line and may switch them to the ZERO state. As shown in Fig. 3 the allowable signal is $0.1V_1$, so attenuation must be added to limit the output signal to a safe value, $0.05V_1$. Then (1) follows:

Maximum Signal

$$= 0.05V_1 = \frac{0.3V_1\left(R_{TL}\left\|\frac{R_1}{n-1}\right.\right)}{R_{TS}+\left(R_{TL}\left\|\frac{R_1}{n-1}\right.\right)+R_2}. \quad (1)$$

The vertical lines mean that the two resistances are to be taken in parallel. R_{TL} is shunted by the unswitched elements on the digit line, which are taken to be in the ONE state with resistance R_1 to ground. R_{TS} and R_{TL} are again the resistances seen from the matrix on the word and digit sides, respectively, as shown in Fig. 4.

Equations relating to Fig. 3 are:

$$I_1 > 10^{-6}A, \quad (2)$$

$$I_1/I_3 < 1/5, \quad (3)$$

$$R_1 = \frac{0.3V_1}{I_3 - I_1}. \quad (4)$$

The manimum value of I_1 is dictated by the properties of the material and is the least designable of the cryosar parameters. Eq. (3) insures a usable ONE region while (4) relates the slope of the line and its end points. Combining these equations gives

$$R_1 < 7.5 \times 10^4 V_1. \quad (5)$$

The SNR due to sneak-path noise may be obtained from the equivalent circuit shown in Fig. 11. The worst case occurs when a ZERO is to be read out while the remainder of the memory is in the ONE state. A word line is pulsed, and all the digit lines but one have ONE output voltages. These are unfortunately fed back up through the matrix and induce a voltage on the nonselected word lines, which in the practical case cannot be kept shorted out. Similarly, these voltages are fed back down through the matrix and induce a voltage in the digit line on which a ZERO is to be detected. An equation for the ONE to ZERO ratio may be simply derived from the figure if it is assumed that each resistive divider does not load the previous divider. Also, n and m are taken to be much greater than one. The ZERO voltage is a direct term from the cryosar in the ZERO state plus a term representing that fraction of the ONE voltage fed up to the word side and back down to the digit line.

$$\text{ZERO} = \frac{0.5V_1R_{TL}}{R_0+R_{TL}} + \text{ONE}\left[\frac{R_{TS}}{\frac{R_1}{m}+R_{TS}}\right]\left[\frac{R_{TL}}{\frac{R_1}{n}+R_{TL}}\right]. \quad (6)$$

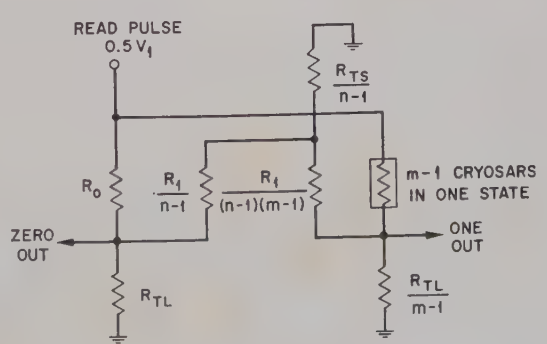


Fig. 11—Equivalent circuit for sneak-path noise.

[4] R. C. Johnston, "A Random-Access Cryosar Memory," presented at Northeast Electronics Res. and Engrg. Mtg., Boston, Mass.; November 15, 1960.

If R_0 is large, the direct term may be dropped. If R_{TL} is made large with respect to R_1/n then the last factor goes to one and all the attenuation is obtained in the first resistive divider. Eq. (6) then gives quite reasonable and unrestrictive values for R_{TS} and R_{TL}. However, there are other considerations which lower the permissible values for these parameters. The word-line impedance R_{TS} must be small so that the load seen by the driver when reading all ONES does not affect its output voltage.

Suppose a transmission line is terminated in its characteristic impedance Z_0 in parallel with a variable load R_L. The voltage at the load will be less than the source voltage as given in:

$$V_L = \frac{1}{1 + \frac{Z_0}{2R_L}} V_S. \tag{7}$$

If R_L is required to be greater than Z_0, the variation in V_L is limited to ± 20 per cent. For the present problem R_L is identified with the resistance looking into the matrix on a word or digit line. The impedance looking into the drive lines from the matrix, R_{TL} or R_{TS}, is $1/2\ Z_0$ for the transmission line case above, since the line appears in parallel with its terminator. Substitution of these identities into the inequality $Z_0 < R_L$ gives

$$2R_{TL} < \frac{R_1}{n-1}, \tag{8}$$

and

$$2R_{TS} < \frac{R_2 + \left(R_{TL} \left\| \frac{R_1}{n-1}\right.\right)}{m-1}. \tag{9}$$

No attempt is made to avoid loading of a digit line by a cryosar in the TWO state. The feed-through of the word pulse may even change the polarity of the voltage on the digit line, but no harm is done as the cryosar is already switched.

The calculation of the power dissipation in the helium is tedius because of the many terms, but it is not conceptually difficult. In addition to the steady-state power P_{ss} the losses under the worst case condition of continuous write must be added to obtain the total power P.

The design procedure is to assume values of R_1, n, and m: calculate R_{TL} from (8); assume R_{TS} [keeping below the upper bound given by (6)]; calculate R_2 from (9); check the signal against (1); calculate V_1 from (5); and finally check the power dissipation against the available cooling power. The results of two such calculations are shown in Table I, in the "double-break, normal" columns. The assumed values not mentioned are signal/noise = 4, minimum line impedance = 2 Ω, $I_1 = 1$ μa. The major portion of the dissipation occurs as dc power when all ONES are stored. The remainder is pulse power during worst-case conditions of continuous write with most of the power lost in the *nm* cryosars subjected to the digit pulses. All pulses are assumed to have a duty factor of $\frac{1}{3}$, with write pulses alternating with clear pulses.

TABLE I
RESULTS OF EIGHT MEMORY DESIGNS

	Double Break		Single Break		Double Break		Single Break	
	Normal	Isolator	Normal	Isolator	Normal	Isolator	Normal	Isolator
R_1	20K	20K	20K	20K	20K	20K	20K	20K
n	1024	1024	1024	1024	512	512	512	512
m	256	256	256	256	64	64	64	64
R_{TL} (8)	10	10 and 100	10	10 and 100	20	20 and 100	20	20 and 100
R_{TS} (6)	<120	<84	<120	<84	<460	<340	<460	<340
R_{TS} (9)	2.5	2.5	40	40	10	10	150	150
R^2 (9)	1250	1250	—	—	1250	1250	—	—
Signal (1)	$1.6\times10^{-3}V_1$	$4.0\times10^{-3}V_1$	$10^{-4}V_1$	$2.5\times10^{-4}V_1$	$4.0\times10^{-3}V_1$	$6.8\times10^{-3}V_1$	$2.5\times10^{-4}V_1$	$4.3\times10^{-4}V_1$
V_1 (5)	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
P_{ss}, watts	9.4×10^{-2}	9.4×10^{-2}	9.4×10^{-2}	9.4×10^{-2}	1.2×10^{-2}	1.2×10^{-2}	1.2×10^{-2}	1.2×10^{-2}
P	0.131	0.166	0.136	0.136	0.015	0.020	0.015	0.015

The calculated power dissipations are reasonable and well within the capabilities of existing or proposed closed cycle helium liquifiers. The lowering of R_{TS} and R_{TL} below the levels demanded by sneak-path noise is unfortunate and results in an output signal much lower than initially hoped for. However, the output is single ended and an extra stage or two in the sense amplifier could produce an adequate signal with good ONE/ZERO ratio.

A circuit that allows R_{TL} to be large when driven from the memory side and small when driving the memory is shown in Fig. 12. A simple, uncompensated cryosar is added on each digit line as an isolator. This scheme enables more freedom in choosing R_{TL} and R_{TS} and gives more signal output. The higher value of R_{TL} should be used in (1), (6), and (9), while the lower value is used in (8). The results are shown in Table I in the "isolator" columns.

Designs using the single-break cryosar are also shown in Table I. R_1 and R_2 are set equal in this case.

The shunt capacitance of cryosar B (Fig. 2) must be low because it causes an overshoot during removal of

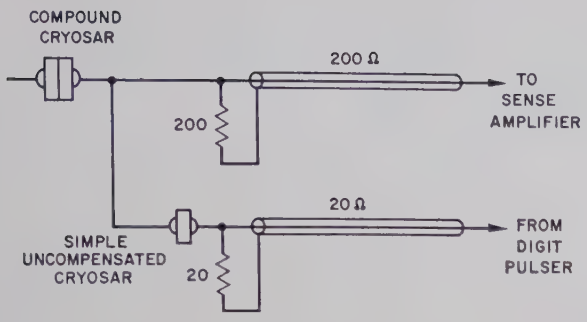


Fig. 12—Isolator circuit for R_{TL} of 10 and 100 ohms.

the read pulse that may reduce the current in the A section of the compound cryosar below the critical level I_1 causing the element to switch to the ZERO state. Given values of a 10 nsec fall time, a maximum current overshoot of 2 μa, and a V_1 of 0.3 v, then the limit on C_B is about 0.13 pf. The same limit also applies to the shunt capacitance of R_1 of the single-break compound cryosar. Note that these are not shunt capacitances to ground, which can be much higher as they are in parallel with the line terminators.

Conclusion

The interest in this device for a computer memory is based on the following considerations:

1) The device may be switched with pulses the order of 10-nsec width.
2) Very high element densities are foreseen as there is no design penalty paid for decreasing the dimensions.
3) Memory planes, complete with wiring, could conceivably be fabricated in one step with good uniformity.

It is impossible to anticipate every difficulty in the building of the first large cryosar memory in such a paper design as this. However, there is sufficient flexibility in the device and circuit parameters to allow the probable success of such a large-scale program.

There are many details which have not been mentioned. If the realization of transmission lines of very low and high impedances proves impractical, transformers or resistive pads will have to be used and the extra power dissipation accounted for. The heat conducted by the transmission lines must be considered also, but this could be reduced by the addition of thermal isolators in the lines.

The calculated values for R_1 and C_B are stringent, but it is felt that they can be attained with the use of thin wafers and stripe densities of hundreds to the inch. Uniformity would be helped by such densities also.

The eight design cases in Table I may be compared by observing the figures for ONE signal. It appears that the double-break gives a signal improvement of 16 over the single-break cryosar; the isolator circuit gives an increase of signal of about two over the normal circuit; and the factor of eight decrease in size of the smaller memories gives a signal increase of 2.5. The reason for the double-break cryosar is now apparent. Of course this increase in signal must be weighed against the added fabrication problems of the double-break cryosar matrix and its probable slower speed in the read operation.

Acknowledgment

The author would like to thank R. H. Rediker and A. L. McWhorter for supplying him with the experimental cryosars.

A Method for Resolving Multiple Responses in a Parallel Search File*

E. H. FREI†, MEMBER, IRE AND J. GOLDBERG‡, MEMBER, IRE

Summary—It is possible to build memories in which the contents of all registers are tested simultaneously, and in which there is a single indication of the presence or absence of any number of positive responses to the test criterion. A method is described for separately identifying the members of a set of responses by presenting sequences of tests which generate an identification number for each member. The testing algorithm is easily mechanized, and the number of tests required per item is approximately proportional to the logarithm of the number of file registers. The method also may be used to search for items with contents falling within arbitrary numerical ranges.

Introduction

THERE HAS been recent interest in memories of special construction, in which an item is found on the basis of part or all of its data content, and in which the search is conducted simultaneously on all items in the memory. Among the adjectives used to describe this structure are "Catalog," "Associative," and "Content-Addressed."[1–5] In many applications there is a possibility that several items will satisfy the search criterion. For simplicity of construction, it would be desirable to provide only a single indication as to whether or not *any* matching items are present, provided there were some means of identifying the responding items individually.

The purpose of this paper is to describe a relatively simple method for resolving the multiple responses of such a memory, or file, using only a single indicator for the entire file. The method is economical in time and storage space, and does not require that the memory be alterable.

File Structure

The file is assumed to be made of a number of identical registers containing binary data. An interrogation is applied to all registers simultaneously, via a set of terminals, one for each bit of the standard register. Each terminal may be driven to one of three states—0 and 1, corresponding to the binary variables in the data registers, and X, the "don't care" state. The file output appears at a single terminal, which assumes two states, indicating whether or not there is at least one register in the memory whose contents "match" the data on the input terminals. In principle, the output indication appears immediately after presentation of the interrogation. There are several possible rules for determining the "match" operation. One simple rule is that of identity between the corresponding bits of the search and data patterns. Another important possible rule is pattern inclusion, *i.e.*, a data register's contents "match" a search pattern if there is no bit place at which the data bit is 0 while the search bit is 1. The appearance of the X state at an input bit position removes that position from any test, thus permitting the test to be made on any desired portion of the data field.

It is further assumed that a fixed part of each register is devoted to a unique serial number which identifies the register. This number is encoded in simple binary code, and it is further assumed, for convenience of explanation only, that there are no gaps in the enumeration. Actually, any coding scheme whatever may be employed. The method to be described will generate separately the serial numbers of the responding items, but the order will be that given by interpreting the patterns as simple binary numbers. In many binary-coded-decimal codes, the numbers will appear in natural order. In any case, there need not be any correspondence between serial number and physical position in the file.

Finally, it is assumed that the test on the serial-number subfield is on the basis of identity, with possible "don't cares," but that the data subfield may be tested by any suitable rule.

In searching for matching items, the user of the memory will first present the search pattern of interest to the "data" subfield, with the "serial-number" subfield entirely set to the "don't care" state. If the output signal indicates that at least one matching item has been found, the user will proceed to identify the serial numbers of the several items by means of a sequence of tests on the serial number subfield, each time repeating the contents of the data subfield.

Rules for Testing Serial Numbers

A desired serial number will be generated bit by bit in a sequence of alterations and tests. There are two modes of searching, Mode A and Mode B. The first item in a set of multiple responses will be found by using

* Received by the PGEC, May 3, 1961. The work reported in this paper was sponsored by the Storage and Retrieval Sect., Rome Air Dev. Ctr., under Contract AF 30(602)-2142.

† Weizmann Institute of Science, Rehovoth, Israel.
‡ Stanford Research Institute, Menlo Park, Calif.

[1] A. E. Slade and H. O. McMahon, "The cryotron catalog memory system," *Proc. Eastern Joint Computer Conf.*, December 10–12, 1956, New York, N. Y., vol. 18, pp. 115–120.
[2] R. R. Seeber, Jr., "Associative self-sorting memory," *Proc. Eastern Joint Computer Conf.*, December 13–15, 1960, New York, N. Y., vol. 18, pp. 179–188.
[3] G. G. Stetsyura, "A new principle for the construction of a memory device," *Dokl. Akad. Nauk SSSR*, vol. 132, pp. 1291–1294; June, 1960.
[4] J. Goldberg, "Binary Tests for Two Terminal, Simultaneous Action Data Retrieval Machines," Stanford Res. Inst., Menlo Park, Calif., Suppl. B to Quart. Rept. 2, SRI Project, 3101, Contract AF 30(602)-2142; 1960.
[5] W. L. McDermid and H. E. Petersen, "A magnetic associative memory system," *IBM J. Res. and Dev.*, vol. 5, pp. 59–62; January, 1961.

Mode A. Each successive item is found by applying one Mode-B sequence followed by one Mode-A sequence. The rules will be described by reference to a serial-number field of $n+1$ binary digits $(d_n, d_{n-1}, \cdots, d_1, d_0)$ with d_0 the least-significant digit. It should be understood that the entire field of digits is tested as a whole, both serial number and data, and that digits which are not mentioned explicitly are assumed to remain in their previous state.

Mode A

At the start of any Mode-A sequence, there will be a continuous string of X ("don't care") symbols, starting from the least-significant digit, *e.g.*, d_5d_4XXXX, which represents the range of numbers d_5d_4 0000 to d_5d_4 1111 (d_5 and d_4 will be given by a preceding Mode-B search). On the first test of a search, all digits will be X.

Mode A proceeds by setting the X digits to 1, one at a time, starting from the most-significant X digit, and testing the file for the resulting pattern, *e.g.*, d_5d_41XXX. If the test is positive, the trial 1 is preserved in the next pattern; if it is not positive, the trial 1 is changed to 0, *e.g.*, if d_5d_41XXX fails, the next test pattern is d_5d_401XX.

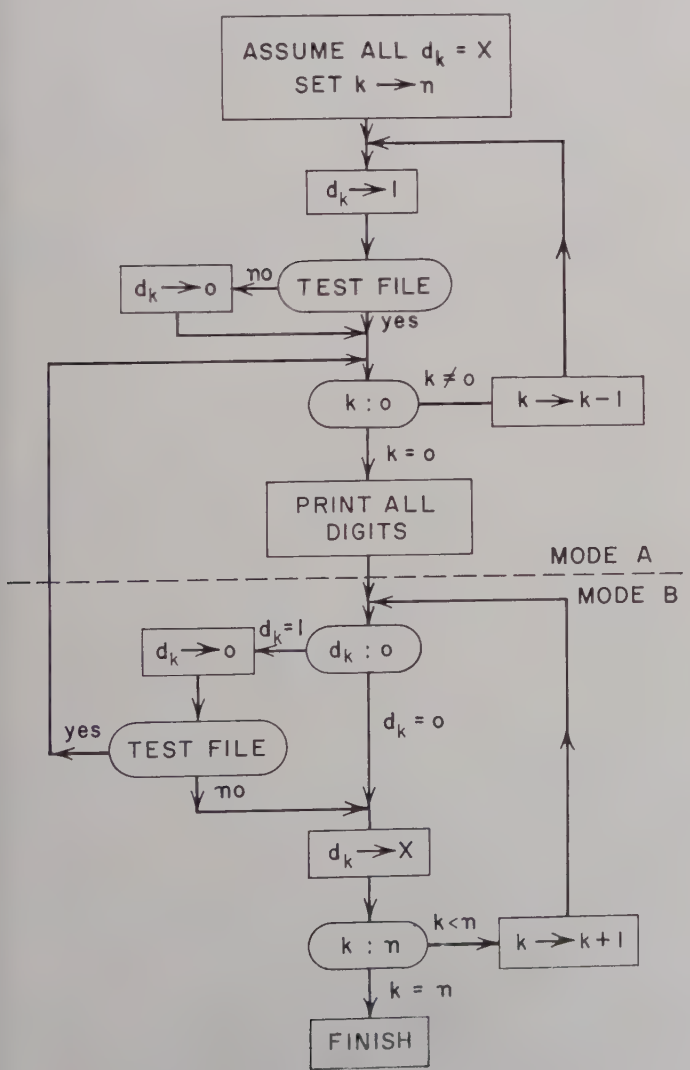


Fig. 1—Flow chart for test procedure.

When all X's have been removed, the resulting number is that of the highest-numbered responding item in the given number range. The testing now proceeds in Mode B.

Mode B

Mode B starts with a complete number which contains no X symbols. In successive steps, it establishes continually lengthening strings of X symbols, starting from the least-significant digit, but each string is headed by a 0 symbol.

On the first test of a B sequence, if the least-significant digit is 1, it is changed to 0, and the resultant pattern is tested. It may be accepted as a new responding item. Whether or not it is accepted, a new B sequence is started. The next test pattern is established by changing the least-significant 1 digit to 0, and all lower digits to X. For example 010110 is followed by 01010X, 1100XX is followed by 10$XXXX$, etc.

If a new B pattern is accepted, a range of numbers containing one or more new responding items has been found, and the testing changes to Mode A. If no new B pattern is accepted, *i.e.*, if, ultimately, 0$XXXXX$ is rejected, the search is ended.

A more exact description of the testing procedure is given in the flow chart of Fig. 1.

The behavior resulting from the rules given in the flow diagram may be visualized more conveniently by reference to the diagram in Fig. 2, which might be called a "decision tree." Here, all the sixteen numbers given by a field of four bits are arranged as terminals of

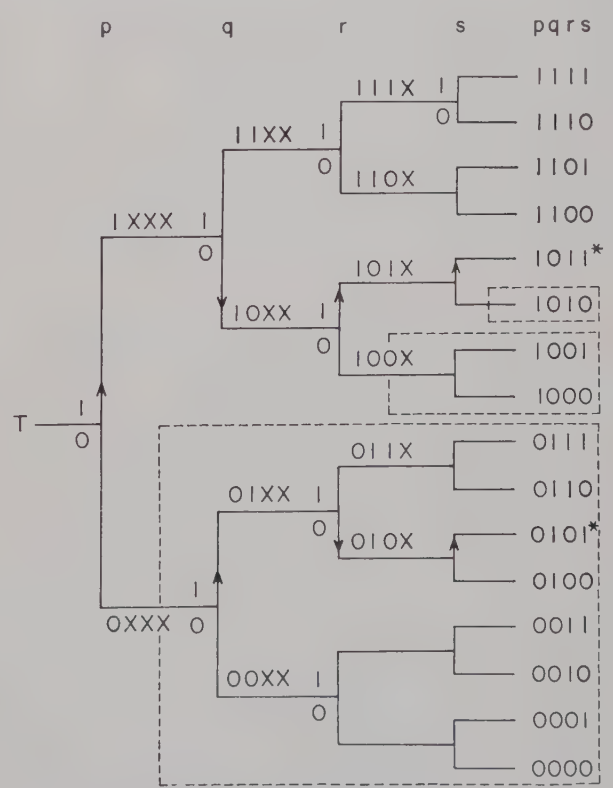


Fig. 2—Decision tree.

a graph. If the upper-branch path at a node is labeled *1* and the lower path is labeled *0*, the number at the right-hand end of a path is given by the sequence of *1*'s and *0*'s corresponding to the labels of the path connecting it to the source node T. The successive levels of nodes are labeled p, q, r, and s, corresponding to the bits of the serial numbers. Several branches on Fig. 2 are labeled in the form $10XX$, which in this case indicates that all numbers having the leading bits "10" belong to the branch.

As an example, assume that there are two responding items, with serial numbers 1011, and 0101, respectively (marked with asterisks). Mode A of the testing rules may be visualized as a trial of successive branches, moving left from the source node T, in which the *1* branch is always preferred.

The first test, at level p, consists of trying the upper branch first, thus the first test pattern is $1XXX$. The resulting positive answer leads to a trial of the pattern $11XX$ at level q. The resulting negative answer indicates that branch $10XX$ must be followed, so that the next test may be $101X$, etc. The arrowheads in the upper section of the tree show the path followed in finding 1011, in Mode A.

After a number is found, Mode B is employed to determine the next lower branch containing a response. The next number range tested is, in this case, the single number 1010, exhausting the branch $101X$ (the successive ranges tested by B are enclosed by dotted lines). Following this, the entire branch $100X$ is tested, exhausting the entire branch $10XX$, and hence, $1XXX$. Thus, the next branch tested is $0XXX$. Here, the number 0101 causes a positive response on the test of branch $0XXX$. Mode A may now be employed to find the largest number in that branch. The branches followed are marked again by arrows.

The sequence of tests for this example is summarized in Table I.

The rules given may be implemented in automatic digital circuitry and, in fact, were quickly set up on a laboratory switching-network tester at Stanford Research Institute, by Dr. B. Elspas.

TABLE I
TEST SEQUENCE FOR FINDING NUMBERS 1011 AND 0101

Test Number	Test Pattern	Match Indication	Mode
1	$1XXX$	yes	A
2	$11XX$	no	
3	$101X$	yes	
4	1011*	yes	
5	1010	no	B
6	$100X$	no	
7	$0XXX$	yes	
8	$01XX$	yes	A
9	$011X$	no	
10	$010X$	yes	
11	0101*	yes	
12	0100	no	B
13	$00XX$ end	no	

General Properties of the Test

Mode A may be recognized as a form of the familiar binary search test commonly employed in searching lists, *i.e.*, the list is halved, the proper half is chosen and halved, and so on, until a point is isolated. In this instance, since responses may occur in both halves of a number range, the higher-numbered subrange is preferred arbitrarily. In Mode A, this preference is accomplished by trying each new subject digit as a 1.

Mode B is novel, and it is essential, since repeated application of Mode A would simply produce the same number. Mode B acts to ignore serial numbers larger than that present at the start of Mode-B test, since it never causes a digit to be changed from 0 to 1. Mode B does not find an explicit serial number but rather determines quickly the next-lower number range (of a size which is an integral power of 2) which contains one or more responses. Application of Mode A to this range then extracts the highest-numbered entry in the range, and so forth. Examination of the example will show that each new trial "number" in Mode B defines and tests a range which is contiguous to the previous range.

The result of successively alternating applications of those two rules is to produce the serial numbers of matching registers in monotonically decreasing order, each number appearing at the end of a Mode-A sequence.

Dual rules A′ and B′ exist, which produce the serial numbers in monotonically increasing order. There is no basis for preferring one order to another unless there is prior knowledge about the clustering of responses at one end of the number range.

Number of Tests Required

The number of tests required to identify the separate responses depends upon the number of responding items and upon the distribution of their serial numbers in the number range. A test is needed for each change of X to 1 in the A Mode, and for each change of 1 to 0 in the B Mode. For an n-bit serial number, if there is only one responding item, n tests will be needed to find the number, and from zero to n tests to determine that there are no more items, for an average of $(3/2)n$ tests. For several responses, the average number of tests per response drops quickly. Study of several distributions suggests that the maximum number of tests for n responding items is $n+1$. Thus the number of tests required to exhaust the file varies with the number of matching items times the logarithm to the base 2 of the number of file items, rather than directly with the number of file items, as in a serially searched file.

It is possible to calculate exactly the number of tests required in the event that all file items respond. Although the situation is unrealistic, the number is of

interest as a lower bound. The number may be found by induction, as follows.

Consider the table of tests in Fig. 3(a). This sequence, labeled S_2, will result from a search on a 2-bit number field in the event that all four numbers respond, but without *a priori* knowledge that all will respond. Fig. 1(b) is the sequence, labeled S_3, required to exhaust a 3-bit field. It may be noted that the sequence S_2 is embedded twice in S_3, as shown schematically in Fig. 3(c), with the addition of the two terms $1X \cdots X$ and $0X \cdots X$, which distinguish the two appearances. The sequence for four bits S_4 again contains two appearances of the 3-bit sequence, each with a "heading." The number of tests on a field of n bits may be expressed by a finite difference equation, obtained by induction from the above illustration:

$$F(n + 1) = 2F(n) + 2,$$

where $F(n)$ is the number of tests on a field of n bits. Using the boundary condition that $F(1)$ is 2, the solution is

$$F(n) = 2^{n+1} - 2.$$

Since the number of items retrieved in this limiting case is 2^n, the average number of steps per item is $F(n)/2^n$, which is about 2, and it is essentially independent of the number of bits.

Fig. 4 is a curve of the average number of tests per response vs the number of responses, for a field of five bits. The points shown encircled are known exactly, *i.e.*, the extreme cases of one response and all responses. The remainder of the points were obtained experimentally, from a number of reasonable, hypothetical distributions of item numbers.

Application to Size Tests and Interval Searching

The testing method described above was presented as a means of searching the serial-number field of a file. If the data field contains numbers (in addition to or instead of names), Modes A and B may be applied for testing the size of such numbers relative to a given reference number.

To determine if there is any file entry some segment of which has a numerical value less than a given reference number, a Mode B search is followed, starting first with an identity test on the given reference number (all of the field except the segment of interest is set to "don't care"). For example, if the reference number is —101—, and there occurs in the file the value —011—, the testing and file responses will be as follows: Mode B: —101— negative, —100— negative, —$0XX$— positive.

The particular value present now may be found using Mode A. If all file items satisfying this test are desired, the serial-number field may now be tested as described previously, each time setting the data field pattern to —$0XX$—. Of course, if there were no file entry with a numerical value less than —101—, the B search would have determined this fact in the usual manner.

To determine if there is a file entry with numerical value greater than a reference value, the dual of Mode B′ is used, *i.e.*, Mode B′ is followed interchanging the roles of 1 and 0. For example, if the reference value is —010—, and —110— is in the file, the test sequence is as follows: Mode B: —010— negative, —011— negative, —1XX— positive.

To determine whether there is a number present which lies between any arbitrary limits (the limits need not appear explicitly in the file) the following procedure may be used: 1) Test for the presence of the upper limit, 2) Generate the next lower number present, using rules B and A as above. If the number found is smaller than the lower limit, the interval clearly is empty. Determination as to whether the number is smaller than the lower limit may be accomplished either by an external arithmetic unit, or by providing a special register in which the lower limit may be stored, and which is interrogated together with the file. In searching for numbers less than the upper limit, the appearance of the number

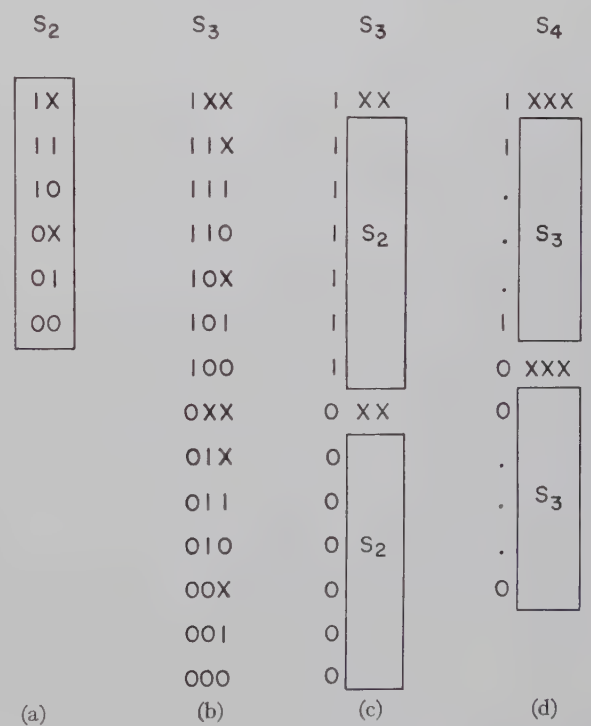


Fig. 3—Test sequences for 2-, 3-, and 4-bit fields all items responding.

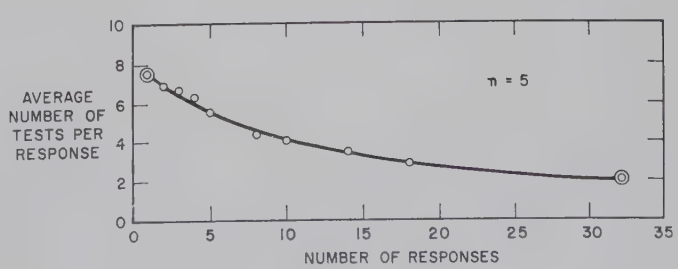


Fig. 4—Average number of tests per response vs number of responses.

held in the special register, *i.e.*, the lower limit, signals the end of the search.

Conclusions and Alternatives

The testing method presented permits efficient and rapid resolution of a set of responses to an inquiry of a parallel-search file in which only a single sensor of presence or absence of matching items is available. The number of tests required is proportional to the logarithm of the number of file items, rather than to the number of items, as in serial searching.

If a bit in each file item were subject to rapid change, the testing method could be simplified. After an item is isolated, the bit could be set, serving to exclude the item from further tests, so that only Mode A would be necessary.

If a number of sensors are used, the file could be subdivided in several ways,[5] but there still remains the choice of whether a file subdivision will be searched in a serial or a parallel mode.

Acknowledgment

The authors would like to thank M. W. Green and R. C. Minnick for their helpful suggestions.

Drum Organization for Strobe Addressing*

GERHARD L. HOLLANDER†, senior member, ire

Summary—**In strobe addressing, a pulse from the memory drum or disk forces the control unit to accept a program word. Strobe addressing for drums or disks costs less than 20 per cent of conventional addressing for real-time control programs, but becomes impractical for very complex problems.**

A comparison of various forms of strobe and conventional addressing shows that simple strobe addressing uses the memory as efficiently as conventional addressing if the problem can and must be solved in one revolution. By interlacing instructions with complete freedom for the program to switch interlaces, a strobe-addressed drum appears to have as many parallel bands as it has interlaces, thereby extending the advantages of strobe-addressing to more complex problems. Only minor changes in the program memory are necessary, and the arithmetic element and control unit remain essentially unchanged.

The irregular interlace organization is described for an airborne computer with an 8-fold interlace, and its effectiveness is evaluated. In a test problem, this method reduces the size of the required memory by a factor of four and increases the effective computing speed fourfold.

I. Introduction

ONE OF THE critical characteristics of a computer is its economy, the work it performs per unit cost. "Cost" is used here in its broadest sense meaning not only the dollars of initial expenditure, but also such penalties as size, weight, maintenance, and unreliability. Two factors determine computer economy: computing speed and the number of components. Broadly speaking, a faster computer can perform more work, but more components make it more expensive. A good computer design aims toward a high utilization of all its components so as to maximize the economy, the work per unit equipment.

This paper describes a method for increasing the effective computing speed of a drum computer with a concomitant reduction of the drum size through the combination of several simple measures that are ineffectual separately. For computers carried in aircraft where each pound of load may represent an additional $500 initial cost,[1] the new memory organization raises the economy of the computer both through increased speed and through lower cost. Although a specific airborne computer serves as illustration, the method can be useful for other drum or disk computers.

The Problem

The program memory of an airborne computer accounts for about half its weight, which makes it a prime target for "cost" reduction, especially if the performance can be retained through unusual organization. For high utilization, without the cost of a large random-access memory, the computer should be programmed for minimum (or zero) access delay. Since, in contrast to scientific and business problems, real-time problems such as aircraft control have a long fixed program but only a few operands (data), it is economical to provide a small random-access memory for the data. Since the program is executed in a largely predetermined order, it could be placed in a single long band if this does not lower the duty cycle of the computer too much. This reasoning led to strobe addressing in the C-1100 com-

* Received by the PGEC, July 22, 1959; revised manuscript received, November 10, 1960.

† Hollander Associates, Consultants, Box 2276, Fullerton, Calif.

[1] G. L. Hollander, "Real-time airborne digital computers," *J. Instr. Soc. Am.*, vol. 5, pp. 29–30; 1958.

puters—a series of parallel general-purpose computers with typical execution times of 15 μsec for addition and 60 μsec for square-root.[2]

How does strobe addressing differ from conventional memory organization? The conventional way to locate a sector on a drum or disk is to count synchronizing (clock) pulses and compare the counter contents with the address register. But the counter, address register, and comparator can be eliminated—without serious programming restrictions on problems coded for minimum access delay—by forcing the control unit to use each order or constant when the memory produces it. The memory slots in this system have no address number; and instead of incrementing a counter, each pulse in the synchronizing track strobes a valid instruction to the control element. To retain the ability to branch, a few control flip-flops can, on command, conditionally disable the control element. The memory is strictly serial by word (though parallel by bit). Alternative routines or other program-segments are selected by performing or ignoring a group of instructions, but the computer waits idly when instructions are ignored. For simple applications, strobe addressing proved to be best; the goal of this study is to extend it to complex problems by eliminating certain shortcomings.

The shortcomings for complex problems can be illustrated with a simplified practical example. Table I summarizes some typical tasks for computers in military aircraft. As any phase of the flight requires fewer than half and often only a quarter of the thirteen routines, computers with a strictly serial program memory are idle most of the time, so that the over-all performance is slow compared to their high arithmetic speed.

As with any noninterlaced serial memory, minimum-latency programming precludes sharing of subroutines; and critical routines to be executed more than once per revolution must be duplicated. These added routines increase the store size and its access time, forcing even more duplication. As more alternative routines (for example, several navigation methods) are added, this cumulative interaction causes the store size and access time to increase at least as the square of the complexity of the problem, so that the basic serial memory approach becomes uneconomical for problems more complex than Table I. Finally, the inability to address a specific location complicates nesting of program segments and precludes utilization of the vacant slots for nonoptimum programs.

Conventional measures,[3] short of using a large random-access memory, provide no solution. Strobe addressing already implies minimum-latency programming. Although, in general, fast-access loops shorten the access time to selected data, in the C-1100 the small (64-word) core memory already fulfills this function. Multiple drums or data bands cost and weigh too much.

TABLE I

Typical Tasks of a Central Computer in an Aircraft

Flight Phase \ Routines	Air Data	Autopilot	Cruise Control	Landing	No Nav. Aids	Vortac	Waypoint	Set	Check Point	Intercept	Weapon Delivery	Operability Test	Diagnostic Check
				Navigation									
Warm Up													X
Preflight Checkout													X
Taxi			X										
Take-Off	X	X	X			X	X	X					
Climb	X	X	X			X	X	X					
Cruise Nav. Aids	X	X	X	X		X	X	X					X
" " " (Way-Point)	X	X	X	X		X	X	X	X				
No " "	X	X	X	X	X	X	X	X					X
No " " (Check-Point)	X	X	X	X	X	X	X	X			X		
Strike	X	X	X	X	X	X	X	X				X	
Defense	X	X	X	X	X	X	X	X				X	
Land	X	X	X	X	X								
Intercept	X	X	X	X	X	X	X	X			X		X

Highlights of the Solution

Serial programs in the same data tracks can be made to appear parallel to the computer, if many properly interlaced instructions are available from the drum in the execution time of one order. Instead of the at-least quadratic increase of storage size and access time with problem complexity in the serial memory, storage increases only approximately linearly and access time remains almost constant through the apparent parallelism.

In a practical problem about twice as complex as Table I, this interlacing can increase fourfold the effective computer speed over simple strobe addressing with a fourfold reduction of the memory size, so that the inexpensive serial memories remain economical. Higher computer speed is attained by allowing the computer a higher duty cycle. A smaller memory suffices as vacant slots and program duplication can be avoided. The arithmetic speed of the computer remains the same; the change in the control section is minor.

Scope of the Paper

The next section describes various methods of strobe addressing and then compares them with conventional addressing on the basis of cost and memory utilization. Section III describes the interlaced strobe address organization. The gain in computer speed and memory size are evaluated in Section IV.

II. Comparison of Address Track Organization

In many computers the memory organization determines the computer economy. In large-scale business systems, most of the cost and time is invested in memory components and reshuffling of data in the memory. In airborne computers approximately half of the weight is in the program memory.

[2] G. L. Hollander, "Transac C-1100: transistorized computers for airborne and mobile systems," IRE Trans. on Aeronautical and Navigational Electronics, vol. ANE-5, pp. 159–169; September, 1958.

[3] D. D. McCracken, "Digital Computer Programming," John Wiley and Sons, Inc., New York, N. Y., pp. 228–230; 1957.

TABLE II

COMPARISON OF MEMORY ORGANIZATIONS

	Memory Organization and Address Method	Memory Characteristics			Amount of Hardware						Memory Cost Factors				Memory Utilization		Relative Cost Per Word in Example	
					Address & Control				Data		Formula		Example					
		No. of Interlaces	No. of Data Bands	Relative Effective Access Time	No. of Tracks	No. of Read Amplifiers	Bits of Address Register and Comparator	Other Control Electronics in Bits of Register	No. of Tracks	Read/Write Electronics (Equivalent Number of Amplifiers)	Address and Control	Data	Address and Control	Total	Minimum-Latency Routines Only	All Routines	Minimum-Latency Routines	All Routines
Conventional	Full Address	1	1	1	n	n	n	0	d	d	$3n$	$3d$	36	96	0.6	0.8	160	120
	Clock Track & Counter	1	1	1	1	1	n	n	d	d	$2n+2$	$3d$	26	86	0.6	0.8	140	110
	Multiple Data Bands	1	m	$1/m$	1	1	n	n	md	$(1+.2m)d$	$2n+2$	$1.4md+2d$	26	264	0.7	0.8	380	330
Strobe	Single Strobe Track	1	1	1	1	1	1	1	d	d	4	$3d$	4	64	0.6	0.6	110	110
	Program Interleave	1	1	1	2	1	1	2	d	d	6	$3d$	6	66	0.6	0.9	110	70
	Instruction Interlace	m	1	$1/m$	m	$1+.1m$	1	$1+\log m$	d	d	$1.1m+6$*	$3d$	15	75	0.7	0.8	110	90
	Selector Tracks	m	1	$1/m$	p	$1+.1p$	1	$1+\log p$	d	d	$1.1p+6$*	$3d$	20	80	0.9	1.0	90	80

* For range of interest $\log_2 m \approx 3 \approx \log_2 p$.

A fruitful area for reducing the required size of the memory is by special addressing means.[4,5] Table II compares the relative costs of three conventional and four strobe-addressed memories with 2^n storage locations (slots) with d parallel bits per data word. (For the sake of clarity, a few secondary effects will be neglected here.) Let us first examine the column headings.

For this comparison, the memories can be characterized by the number of interlaces, the number of data bands, and their relative effective access time. The interlace is the ratio of the execution time of a short[6] instruction to a memory word interval. For low cost and low weight, all information should be in a single data band, especially when the speed of the computer demands parallel storage. The relative effective access time measures the number of memory words available during the execution of one short instruction. The relative effective access time can be reduced either by multiple data bands or by interlacing at a higher clock rate.

The key cost items consist of the tracks and associated hardware, the read-write electronics, and the additional electronics for control. Control electronics include the address register, the comparator, and associated counters and switches. The data hardware columns pertain to a parallel data organization of "d" binary digits per word. The relative costs for the address-and-control portion and for the entire store are based on the unit cost of the read/write electronics in the last hardware column being double the unit cost of the other columns. The numerical example is a 4096-word drum ($n=12$) for a 20-bit machine ($d=20$) using an 8-fold interlace ($m=8$).

The size of the memory is determined by the memory utilization. Here the portion of memory useful for minimum-latency programming must be distinguished from the total utilization including the portions that could not be minimum-latency programmed. The minimum-latency column shows the utilization by programs in which almost all instructions and constants are available from the memory (without time delay) when the computer needs them. The total utilization is the practical maximum when as much of the drum as possible is used for minimum-latency programs.

The last two columns compare the relative cost per word of the minimum-latency program and per word in the memory. They are computed by dividing the total memory cost in the representative example by the appropriate utilization. The lower relative cost per word of strobe addressing will show up later as one of the factors in the improved economy.

Conventional Approaches

The first three rows describe the conventional approaches to addressing a drum. The full address is stored only where special circumstances justify the high cost. For example, each of the 4000 slots on a drum may carry the 12-binary-digit designation of its address; but 11 tracks can be saved by using a single clock track with an associated counter that contains the present address. The counter and clock-track approach predominates with either single or multiple data bands.

To achieve almost zero latency, the conventional methods must use about one quarter of the memory either for jump instructions or for the "next-word" por-

[4] G. L. Hollander, "Quasi-random access memory systems," *Proc. of the Eastern Joint Computer Conf.*, vol. 10, pp. 128–135; December, 1956.

[5] G. L. Hollander, "Addressing System for Data Records," U.S. Patent No. 2,765,456; October 2, 1956.

[6] Any instruction that can be executed in one add time, such as add, compare, transfer, jump. Examples of long instructions are multiply and divide.

tion of the instruction. Furthermore, the entire drum cannot be used for minimum latency programming. After more than half of the slots are occupied, the choice is too limited to guarantee minimum-latency for any additional instructions.

Multiple data bands reduce the effective access time and are the conventional, although expensive, way of achieving the desired goals. If the data bands can be switched between successive words, queueing theory or intuition shows that a larger portion of the drum can be used for minimum-latency programs; but microsecond switches for read and write currents are expensive.

Strobe Tracks

Since with minimum-latency programming, an instruction is available when the control element is ready for it, we can simplify the computer by forcing the control element to accept an instruction when the memory presents it. This replaces the n bits of address counter, address register, and comparator with a single flip-flop that can conditionally shut off the stream of instructions.

The control section accepts an instruction from the program drum whenever a pulse appears in the clock (strobe) track. Under normal conditions, the computer carries out one instruction after the other, as each appears on the drum. Inasmuch as it takes three to four times longer to execute a long instruction than a short instruction, several slots are left vacant after each long instruction, so that the next instruction is ready when the previous one has been completed.

The computer does not directly address an instruction; for conditional and unconditional jumps the computer is shut off until it again reaches valid instructions. Thus, subroutines are selected by performing or jumping over a set of instructions. Even though a group of instructions is jumped, the computer must wait for the next set, just as if it had not jumped.

The virtue of strobing is low-cost addressing—about one-sixth of the clock track and counter—but memory utilization is low. Vacant slots after the long instructions (about 40 per cent) are usually wasted, because to nest part of another program takes at least two control orders and is awkward even then. Nesting of alternative routines does not pay for fewer than four or five consecutive vacant slots. Strobe addressing can be as efficient as the conventional methods for the minimum-latency portion of the program, but the conventional methods can pick up for non-minimum access programs the remaining slots that are wasted with a single strobe track.

Program Interleaving[7]

About 70 per cent of the vacant slots can be used for a second program controlled by a second strobe track

[7] G. L. Hollander, "Transac C-1100," *op. cit.*, p. 166. Program interleaving was conceived independently by C. Ramamoorthy of Minneapolis-Honeywell and used in their C-1102.

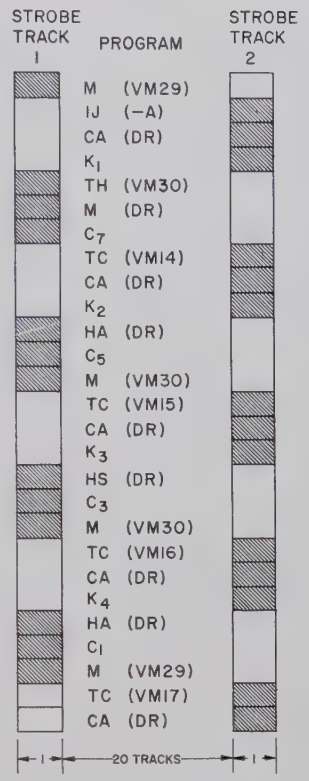


Fig. 1—Interleaving two programs.

(Fig. 1). With a fairly consistent mix of short and long instructions, this second program usually approaches minimum-latency. Besides reducing the required memory size by eliminating vacant slots, the computer duty cycle is raised, as jumped instructions occur when the computer is busy instead of making it idle.

In a 20-bit machine, a second strobe track adds less than $\frac{1}{2}$ per cent to the memory cost; but it increases the storage utilization from 60 per cent to nearly 90 per cent, a 50 per cent gain. At this better utilization, the smaller memory needed for a given problem has a lower access time, an important effect neglected in Table II but covered in Sections IV. It should be noted that in program interleaving the memory word interval equals the execution time of a short instruction.

Instruction Interlace

While in program interleaving the first program had no access delay and the second merely picked up the vacant spaces; in interlacing, instructions for different programs are deliberately and regularly alternated. This conventional interlacing pays off when the memory word interval is several times shorter than the execution time of a basic order. Instruction interlacing is possible without increased memory speed, but it is the lower effective access time due to the shorter memory time that permits the better utilization for optimum programs and justifies the additional control equipment.

Table II shows that m-fold instruction interlacing requires m address tracks and additional switching equip-

ment roughly proportional to $\log_2 m$. Otherwise the cost is the same as for the single strobe track, except that the m-fold memory clock rate raises the cost of the memory electronics by an amount small compared to the memory cost, and neglected here.

Each routine is assigned an interlace, with several short ones sharing the same, and an extra long one taking several. Interlaces are scanned in a sequence determined by the need of the program; each interlace under control of a strobe track. An interlace jump instruction makes a common subroutine in one interlace accessible from programs in other interlaces. The need for jump instructions reduces memory utilization over program interleaving, but the program itself can be cut by 50–75 per cent through less duplication of routines and subroutines at the shorter effective access time.

Without the ability to jump interlaces, the access time of the memory would remain 1, instead of $1/m$. The need for extra instructions to jump interlaces prevents the memory from appearing parallel for short program segments.

Selector Tracks

The advantages of program interleaving and instruction interlacing can be combined by associating each strobe track with a routine, rather than with an interlace.[8] These selector tracks can pick up instructions from any interlace without a jump instruction, thus removing all nesting restrictions and providing high memory utilization. The memory now appears as if it had m parallel bands.

For programming convenience, there are usually 20 per cent to 50 per cent more selector tracks p than interlaces m. The practical range of interlaces m or selector tracks p appears to be between 4 and 16, so that their logarithms can be approximated by 3 in the cost factor.

Discussion of Table II

Table II provides only limited information on the relative economy of various addressing methods. The performance of the computer cannot be evaluated until additional data in Section IV are considered. Here we compare only the efficiency of the memory itself.

Table II shows that a single strobe track is the least expensive addressing method for applications in which data are only a small fraction of the entire program. Although the additional selector tracks increase the cost of the address section, the relative cost per word is reduced over the single strobe track, because now so much more of the memory can be utilized. Program interleaving has a lower cost per word than selector tracks, but Section IV will show that the computer is utilized better with selector tracks for greater over-all economy.

[8] Conceived jointly with Mrs. Kathe Jacoby of the Philco Computer Lab.

It might be well to compare any of these drum schemes to a random access core memory. Although the core memory could be fully utilized and would ensure full computer utilization, the smaller core memory still cost five times more than the appropriate drum or disk.

III. Description of Interlace Organization

Fig. 2 shows the new program memory organization. Instructions are placed in any vacant slot in the instruction band regardless of the interlace, with the sole restriction that two instructions for the same program must be separated by at least enough time to execute the first instruction.

Each major routine has its own selector track that selects the appropriate instructions from the instruction band [Fig. 2(a)]. While for programming ease the first few routines are usually written for specific interlaces, shorter routines can be placed into any remaining vacant slots regardless of the interlace. Subroutines that occur frequently can be placed into a single interlace accessible to any of the selector tracks.

For example, many routines in Table I require sine and cosine subroutines, a typical total of about 50. For such programs one interlace, $\frac{1}{8}$ of the drum, can be filled with a continual succession of sine-cosine subroutines so that it is never far to the start of the next one. In the typical problem, the drum contained 15 sine-cosine subroutines, resulting in an average access time with a speeded-up drum of 200 μsec, a permissible delay. The sine and cosine instructions were reduced from $\frac{1}{3}$ to $\frac{1}{8}$ of the total instructions. This action alone saved 30 per cent of the memory capacity with only a small increase in program execution time.

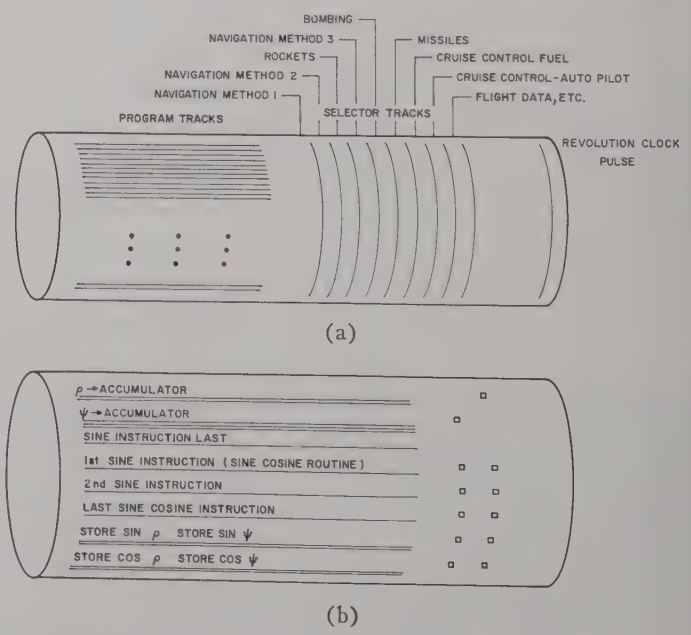


Fig. 2—Proposed interlaced drum organization. (a) Arrangement of program and selector tracks. (b) Detail of above showing how two different routines use same subroutine.

Fig. 2(b) details a small section of the drum in Fig. 2(a) and shows how two interlaced routines can share the same trigonometric subroutines. Depending on which selector track controls the program, either ψ or ρ is transferred in the accumulator; and then the same sine instructions take over. Note that the last instruction from the previous subroutine had not been selected. At the end of the subroutine, the computed values are stored and the individual routines take over. Although not shown in Fig. 2(b), another instruction prior to the trigonometric routine set the index register so that the resulting sine or cosine values can be transferred to different storage locations.

IV. Evaluation

Interlacing reverses the cumulative interaction of increasing store size and access time described in the Introduction for strobe-addressed and other serial memories. Through the apparent parallelism and better utilization by interlacing with increased speed, the access time is lowered, which permits sharing of routines, which reduces the size and access time of the memory further. To evaluate the over-all gain, the separate contribution from each factor is examined and then the combined effect is discussed.

For clarity, we will use the following terms:

Program: The entire set of instructions needed for a given mission.

Routine: The set of instructions required to carry out a given task. For example, in Table I, fuel management, landing, and diagnostic routines.

Subroutine: A set of instructions which recurs frequently in the program or routines; for example, a sine-cosine subroutine.

Analysis of Contributing Factors

Table III summarizes the effect of various actions on the key parameters that determine the computer economy. The reference point for each row is the basic computer with a single strobe track as described by the author.[2] The combined effect follows from combining the values in each column. Let us first examine the individual entries in the table.

Relative values are shown in the table, because specific values depend on the application, and the complexity of exact relations obscures the significant points. Approximate quantitative values for the qualitative figures are shown below the table. A plus sign denotes an increase in the quantity; a minus sign, a decrease. While a minus sign is favorable in every column, the final result is specified completely by the required memory capacity and the program execution time, as they include the effects of the quantities in the other three columns.

The goal is to execute *a given program* in the shortest time with the smallest memory. The three intermediate columns aid in gauging the effect of each factor on the two key columns. How are these columns related to each other? The *memory access time* usually varies directly as the *required capacity*, because at fixed data rates a smaller memory has a shorter access time. The only exception is at the higher drum velocity where the data rates change. The *access time to individual routines* is usually directly related to the *memory access time*. Any difference between these two columns will be covered when it arises. *The time to execute a routine* consists of the execution time of individual instructions and of the access time to the subroutines, and is independent of the *access time to the routine*. The *program execution time* is the sum of *access time to the routines* and *routine execution times*.

Program Interleaving: Program interleaving reduces the *required memory capacity* by utilizing vacant slots after long instructions. It also tends to reduce *access times to routine* further by interleaving long routines, but usually not enough to make the decrease in the *access time to routine* significant.

Higher Drum Velocity: Higher drum velocity does not affect the memory size but reduces its access time—the main cause for repetitive subroutines and routines.

TABLE III

Evaluation of Interlace Effectiveness Relative to a Computer Using a Single Strobe Track

Contributing Factors \ Effect* On	Memory		Computer Performance			Not Used Alone Because:
	Required Capacity	Access Time	Access Time to Routine	Routine Execution Time	Program Execution Time	
Program Interleaving	−1	−1	−1	0	−1	
Higher Drum Velocity	0	−3	−3	−3	−3	Without interlace, computer cannot keep up
Instruction Interlacing	0	0	0	+2	+1	Disadvantageous without increased drum velocity
Shared Subroutines	−2	−2	−2	+2	+1	Routine execution time too long
Elimination of Routine Duplication	−2	−2	+1	0	−1	Access too long to critical routines
Combined Effect	−3	−3	−3	+1	−3	

* Key: +2 = Significant Increase (100 per cent); +1 = Small Increase (20 per cent); 0 = Unaffected; −1 = Small Decrease (20 per cent); −2 = Significant Decrease (50 per cent); −3 = Large Decrease (80 per cent)

All other times are reduced proportionately, including the instruction interval which is now too short for the computer. Without interlacing, higher drum velocities provide no gain if the data rates must be kept low for the computer to keep pace. With interlacing, only the read and write electronics need to be changed.

Instruction Interlacing: Instruction interlacing alone does not improve memory utilization. Without higher drum velocity, the *routine execution time* would increase because the computer must wait for successive instructions. To offset the increased data rate from the higher drum speed, the interlace should increase the instruction interval by +3, which results only in an increase in *routine execution time* as the access time to subroutines is constant. The *access time to a routine* would be unaffected, and the *program execution time* would be longer.

Shared Subroutines: Real-time problems contain many common subroutines, particularly trigonometric conversions and servo-loop control. If these recurring subroutines could be shared, the memory would be much smaller. However, sharing of subroutines on the serial program memory increases the *routine execution time* prohibitively. Common subroutines actually reduce the *access time of the memory*, but this reduction by itself is not large enough to keep the increase of *routine execution time* within tolerable limits.

At the increased drum speed, the *memory access time* is reduced so much that many subroutines can be shared, producing the benefits without the objectionably longer *routine execution time.*

Elimination of Routine Duplication: Some routines, such as the autopilot routine in Table I, must be performed more than once during a drum revolution if the drum revolves too slowly; and every routine duplication increases the memory size, increases its access time, and imposes the need for even further duplication. Although elimination of this duplication reduces the *required memory capacity* and *program execution time*, the key factors, the computer would fail to execute critical routines often enough. Here, again, the increased drum speed comes to our rescue by reducing the *access time of the memory* enough to permit elimination of the duplication.

Combined Effects: The interlace strobe method combines all factors analyzed in Table III. As shown on the last line, the key benefit is the greatly decreased *memory access time.* The critical parameters, *required memory capacity* and *program execution time*, go down by large factors. Only the *routine execution time* goes up slightly, because even at the increased drum velocity the shared subroutines are not always available without delay. However, this is amply compensated by the reduction of the *access time to the routines*, so that the goal of executing a program in shorter time with a smaller memory is attained.

It is interesting to compare the system results in this section with the comparison in Table II. Memory utilization in Table II accounts only for slots left vacant or occupied by unproductive housekeeping instructions. The big reduction in Table III stems from elimination of repetitive routines or subroutines. The faster execution of the program stems from the faster access to the routines and subroutines, which reduces the time that the computer is idle. Required memory capacity and idle time cannot be reduced below equivalent values possible with a random access memory, values they can approach but never achieve. The economy is significantly better than with a random access memory, because of the lower cost per word on drums or disks.

A Typical Problem

A practical problem whose description is beyond the scope of this paper provided a numerical comparison. This problem is more than twice as complex as the example in Table I. The basic C-1100 would have required over 20,000 words of program storage and would have been idle about 80 per cent of the time. Multiple selector tracks and an 8-fold interlace reduced the program storage to fewer than 5000 words; and the idle time to about 15 per cent. This represents a fourfold increase in the effective computer speed with a fourfold reduction of memory size.

The actual savings due to interlaced strobe addressing depend on the complexity of the problem. A simpler problem would yield smaller returns. A problem much more complex than the one above would probably make a random access memory more economical. Thus the fourfold increase in computer speed and fourfold reduction in memory size appear to be the practical limits for real-time problems.

V. Conclusion

Most computers use either random access (core) or sequential (drum) memories for program storage. For many simple control and business applications, drums are cheaper and require less control circuitry. However, the advantage of a sequential memory drops sharply with increasing problem complexity, especially with many long alternative routines. Required storage capacity and access time increase at least as the square of the complexity of the problem, so that the conventional drum approach becomes useless when the computer idles much more than it works.

Interlace strobe addressing retains the economy of a drum for problems with many alternative routines by sharply reducing the normal computer idling time and excessive increase of program storage through:

1) reduction in access time to subroutines,

2) sharing of common subroutines by several routines,
3) minimizing the need for duplicate routine storage, and
4) utilization of normally vacant storage slots.

The gain from each of these steps increases the available benefits of the other three, so that the cumulative effect from their combination is much greater than the sum of the individual gains. In contrast to the quadratic increase of both storage capacity and access time with problem complexity, random interlace selection keeps the storage requirement increase approximately linear and the access time almost constant.

While this interlace method has been introduced with a military aircraft application, it is just as applicable to industrial control and multiple-routine file-maintenance problems. When applied correctly, interlace strobe addressing can utilize the computer almost as efficiently as a random-access memory. Of course, use of a drum must provide enough advantages to offset the slightly greater utilization of the computer by a random-access memory, a utilization which it can approach but never surpass.

VI. Acknowledgment

The author is indebted to Mrs. Kathe Jacoby of the Philco Computer Laboratory for programming several test problems for the strobed interlace method. Acknowledgment is also made to the Philco Corporation and the Hughes Aircraft Company for editorial support and for release of material prepared in their employ.

Computer Languages for Symbol Manipulation*

BERT F. GREEN, Jr.†

This paper is reprinted from a special issue on Automation of Human Functions, published by IRE Transactions on Human Factors in Electronics, *March, 1961. It is an excellent tutorial paper on a subject that should be of interest to all readers of* IRE Transactions on Electronic Computers, *many of whom will not have seen it in its original publication. We are grateful to Dr. Green, Dr. Jerome I. Elkind,* Editor *of* Human Factors in Electronics, *and Dr. Thomas Marill,* Guest Editor *of the special issue for reprint permission. Attention is also invited to the several other interesting papers on "artificial intelligence" which appeared in the same issue.* The Editor

Summary—**Complex, flexible, computer programs can be written easily in list-processing languages. Storage registers are linked together in arbitrary sequences to form lists and list structures, which are the units of the languages. Special provisions are made for recursive subroutines and for hierarchical programs. These particular languages have been used to write game-playing, problem-solving, and other "intelligent" programs.**

DIGITAL computers have outgrown the problems for which they were designed. Simple but tedious mathematical calculations are now only a small part of a computer's repertoire. Complex mathematics, business records and inventories, control of man-machine systems, and "intelligent" automatic systems are all jobs for digital computers today. Computer programs for these problems, however, are very difficult to express in terms of the built-in language of the computing machine. It has become necessary to develop new programming languages that are particularly suited to the more complicated problems. For each such language there must be a special computer program, called a compiler or an interpreter, that translates the new language into the machine language of the computer. Operations requiring large amounts of computation can then be expressed in a small number of words of symbols. Moreover, complex computer programs can be expressed in units that are convenient for the programmer, and not in the units imposed by the machine design.

The first languages to be developed were in the field of mathematics. FORTRAN, IT, and similar algebraic

* Received by the PGHFE, December 9, 1960.
† RAND Corporation, Santa Monica, Calif., on leave from Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass.

compilers permit complex mathematical expressions to be stated clearly in a direct manner.[1] Many scientists who could not take the time to learn machine programming are able to put their problems on a computer by means of FORTRAN or one of its relatives. More recently, it has become clear that similar languages are required for problems that are essentially noncomputational in nature. Business problems of inventory and bookkeeping, for example, require the computer to maintain and sort large files of information. COBOL[2] and FLOW-MATIC[3] have been designed to cope with problems of handling business data. In the field of linguistic processing, which includes automatic language translation, COMIT[4] has been designed to allow the words in the text to be the units that the program processes.

The use of computers in simulating intelligent behavior and in making automata has created the need for another class of information-processing languages. A typical program for a problem in "artificial intelligence" has the following special characteristics:

1) The program involves manipulating symbols that have non-numerical meaning.
2) The storage requirements of the program cannot be specified in advance; complex data structures are developed as the program proceeds.
3) The relations among the elements of the data are restructured while the program is operating.
4) The problem can be described at several levels of detail and the program is stated naturally in hierarchical fashion.
5) The program will be modified often.

There are at least three information-processing languages: IPL, the language designed by Newell, Shaw, and Simon,[5] FLPL, the FORTRAN list-processing language designed by Gelernter and his associates,[6] and LISP, the list processor designed by McCarthy and his co-workers.[7] This paper will discuss the common properties of these and similar languages.

[1] J. Backus, "Automatic programming: properties and performances of FORTRAN systems I and II," *Proc. Symp. Mechanisation of Thought Processes*, Natl. Phys. Lab., Teddington, England, Her Majesty's Stationary Office, London; 1959.

[2] The COBOL system was developed in 1959 by a voluntary committee of government users and computer manufacturers. A language based on this system is described in "The COBOL Translator," IBM Corp., White Plains, N. Y.; October, 1960.

[3] G. N. Hopper, "Automatic programming; present and future trends," *Proc. Symp. Mechanisation of Thought Processes*, Natl. Phys. Lab., Teddington, England, Her Majesty's Stationary Office, London; 1959.

[4] V. Yngve, "A programming language for mechanical translation," *Mech. Translation*, vol. 5, pp. 25–41; July, 1958.

[5] A. Newell and F. Tonge, "An introduction to information processing language V," *Commun. Assoc. Comput. Mach.*, vol. 3, pp. 205–211; April, 1960.

[6] H. Gelernter, J. R. Hansen, and C. L. Gerberich, "A Fortran-compiled list-processing language," *J. Assoc. Comput. Mach.*, vol. 7, pp. 87–101; April, 1960.

[7] J. McCarthy, "Recursive functions of symbolic expressions and their computations by machine, Part 1," *Commun. Assoc. Comput. Mach.*, vol. 3, pp. 184–195; April, 1960.

Lists

The major innovation of list-processing languages is the organization of computer memory. Normally, when a set of symbols is to be stored in a computer, a block of consecutive storage registers is set aside and the symbols are stored in successive registers. If the exact number of symbols to be stored is not known at the time the computer program is written, then enough storage must be set aside to accommodate the largest possible set of symbols that is anticipated. Additional blocks must be set aside for sets of numbers generated by the program. In complex programs, storage allocation can be a major difficulty. The concept of list memories was designed to overcome such difficulties. In a list, each symbol is stored in a separate computer storage register, but the registers need not be consecutive. Each register contains two items of information: 1) the symbol being stored, and 2) the location of the storage register containing the next symbol on the list. This second item of information, called the *link*, serves to keep the list of symbols intact, no matter what particular storage registers are used for storing the symbols. To gain access to the symbols on a list, the programmer needs to know only the address of the beginning or *head* of the list. Each symbol on the list is then located by means of the link of its predecessor. The link of the last register on the list contains a special termination symbol, say O, to signal that there are no more symbols on the list. The arrangement of a list memory is depicted in Fig. 1.

Lists have several valuable properties that stem from using links, rather than sequential storage registers. For example, a symbol may be inserted at any place on the list by merely altering two links. This manipulation does not disturb the other symbols. Similarly, symbols can easily be deleted from lists or moved from one list to another, merely by manipulating the linkages. In order to add symbols to lists and to form new lists, the list processor will need a group of available storage cells. These cells are most conveniently organized as a special "available space" list. Registers are taken from this list as they are needed for storing symbols; other registers are put back on the list when they are no longer required for particular storage needs in the program. Fig. 2 shows how a new symbol is added to a data list, using a cell from the available space list.

A list of symbols may represent a more complex arrangement of data, because each symbol on the list may name a further list. (Each symbol generally has a prefix of some sort, to indicate whether or not it is to be treated as naming a further list. The name of a list is the address of its head.) A bridge deal, for example, might be represented as a main list with four sublists. Each sublist would be the list of cards in one player's hand, while the main list would consist of the names of the four sublists. The order of the four symbols on the main list could indicate the order of bidding and play, *e.g.*, N, E, S, W.

Address of Storage Register	Stored Symbol	Link
473	S1	485
485	S2	631
631	S3	582
582	S4	0

Fig. 1—List of four symbols starting in 473.

Data List

Address of Storage Register	Stored Symbol	Link
473	S1	485
485	S2	~~631~~ *367*
631	S3	582
582	S4	0

Available Space List

371	0	~~367~~ *574*
367	~~0~~ *S5*	~~574~~ *631*
574	0	618
618	0	432
.	.	.
.	.	.
.	.	.

Fig. 2—Symbol S5 is inserted in data list between symbols S2 and S3 using an available cell. The old links have been crossed out, and the new links have been italicized. After the change, the cells on the data list are, in order, 473, 485, 367, 631, and 582. The cells on available space are, in order, 371, 574, 618, etc.

Interchanging E and W on the list would then correspond with East and West exchanging hands.

The complexity can be extended indefinitely, because each sublist can include symbols referring to still further sublists. The cards in a bridge hand, for example, are naturally sorted into suits; the corresponding computer representation would have a sublist for each suit. The bridge deal would then consist of a main list containing the names of the four hands, while each hand would be a list containing the names of up to four sublists, one for each suit. In a hierarchy of this sort, the term *level* is a natural way to distinguish among the various sublists. The bridge deal is at the highest level, the hand at the next level, and the suit holdings at the third level. A main list, together with its sublists, their sublists, and so on, is called a *list structure*. The unit of information in a list-processing language may be a symbol, a list, or an entire list structure.

One data structure, the *attribute list*, is frequently used to characterize a particular list of symbols. In the bridge example, each hand has such attributes as the Goren point count of the hand, the number of quick tricks, the biddable suits, and perhaps others. Since the bridge-playing program will refer to these attributes often, the attributes of each hand are computed once for all, and placed on an attribute list associated with that hand. The attribute list becomes part of the total list structure, so that a bridge deal now consists of a main list containing the four symbols for the hands, a list for each hand, the attribute list associated with each hand, and the sublists containing the various suit holdings for each hand. In more complex programs, both the sublists and the attribute lists can include symbols that name further lists, and any list may have an associated attribute list.

The computer will sometimes be required to build up list structures for itself. In a bridge-playing program, for example, the computer may have to deal the cards, building up the list structures for a bridge deal from an initial list of the fifty-two cards in the deck. Also, if the computer is to *learn* bridge, it will have to keep records of the rules that it uses in various situations, together with the success or failure following the application of the rules. For example, the program might initially have a large number of possible rules to use in selecting an opening lead. As the program played games, its experience might be represented by a list of the good rules for finding opening leads. The list might indicate priority, with rules being moved up on the list when they are successful and being demoted when they lead to disaster. Of course, a very sophisticated machine would keep several priority lists of opening leads, with descriptions of the situations in which to use each.

Processes

In conventional programming, the basic element of information is a digital number stored in a register. The available operations include ADD, SUBTRACT, STORE, and similar numerical manipulations. Machine-language programs combine these operations in sequences. In algebraic compilers such as FORTRAN, the elements of information are constants and subscripted variables each representing a number or a block of numbers; processes are available for algebraically combining these quantities in standard ways. Similarly, list-processing languages include some basic processes for manipulating symbols, lists, and list structures. Processes are provided for comparing symbols, locating a symbol on a list, deleting a particular symbol from a list, combining two lists into one, adding a symbol at the end of a list, etc. A list-processing program is a sequence of these processes. The various list-processing languages differ in the basic processes that are provided for the programmer to use, and in the symbols that the programmer must use in expressing his programs for the processor.

Subroutines are very helpful in conventional programming, and they are indispensable in list processing. A list program consists of instructions for obtaining the inputs

to subroutines, for executing the subroutines, and for storing the results. In a complex program, the subroutines are complex, but the main program is relatively simple. Each subroutine represents a unit of processing, *i.e.*, a block in the flow diagram, while the program represents only the lines of flow. But each block is itself a set of smaller blocks—subroutines—strung together by connections—program. The program can be subdivided into as many stages, or *levels*, as is convenient. Generally, the more levels the better, so that each level can be simple. A list program, then, is a hierarchy of subroutines, successive levels getting more and more detailed. In the bridge example, the highest level would include routines such as "deal the cards," "conduct the bidding," "play the deal," "score the deal." The block "deal the cards" would contain sub-blocks such as "select a card at random," "place a card on a hand." The sub-block "place a card on a hand" would include lower level routines such as, "find the list for the matching suit," "start a new suit list." Still lower we might find "are card X and card Y of the same suit?" List-processing languages are especially designed to keep track of hierarchies of subroutines, so that the program's course through subroutines within subroutines can always be unravelled.

Recursions and Push-Down Lists

List structures may have many levels of sublists, and routines for handling list structures will not generally know how many levels are involved in a particular structure. Routines must be built in a general way to handle any number of levels. One convenient way to achieve generality is to use a *recursion*. Suppose that we are writing a routine to perform a particular process on all the lists of a list structure. The process is first performed on the main list, then on every sublist named on the main list, then on every sublist named on a sublist, etc. Every symbol on the main list that names a sublist can be treated as if it named a list structure; in fact, by the definition of a list structure, it does. But we are preparing a routine to perform a certain process on a list structure, so whenever we encounter a symbol that names a sublist, and therefore a list structure, we would like to execute the routine that we are now writing on this newly named list structure. That is, the routine that we are writing wants to execute itself as a part of itself. Suppose, for example, that our routine is designed to substitute one symbol for another given symbol wherever the latter occurs on a certain list structure. Our routine, which is diagrammed in Fig. 3, will first examine each symbol on the main list. If the symbol matches the given symbol, then the substitution is made. If the symbol on the main list names a sublist, then the present routine is executed with that symbol as the name of the list structure involved. It is assumed in this example that the given symbol does not name a sublist on the list structure. When all of the symbols on the main list have been treated in this way, the process is finished, since all of the sublists will have been examined in the process above. A routine that uses itself as a subroutine, in this manner, is called a recursion.

Recursions are not possible in conventional computer coding. Each conventional subroutine contains a number of storage cells for its inputs, for temporary quantities during its execution, and for the place where the main program is to be resumed when the subroutine is finished. These registers constitute the context of the subroutine. Whenever the subroutine is executed, a new context is set up, and in conventional practice a new context completely destroys any previous context. A recursive routine must save its previous context before performing in a new context, so that when it finishes in the new context, it can revert to the middle of itself and proceed. Special provisions are made in list-processing languages for saving contexts in recursions.

The device used by recursive routines to save context is called a *push-down list*. For every storage cell containing either a parameter or an intermediate result, *i.e.*, every storage cell containing a part of the context, there is a push-down list on which the previous contents of the cell are saved. On a push-down list a new symbol is always added at the top; likewise, symbols are always removed from the top of the list, in the manner of a well for stacking plates in a cafeteria. When a new symbol is added at the top of the list, all of the symbols beneath it are pushed down. As symbols are removed, they expose the symbols beneath them. Thus, the symbols forming the new context of a recursive routine are stacked on top of the previous symbols. At the end of the routine, the present context is discarded, revealing the previous context. In the example in Fig. 3, the first and last blocks keep track of the context by manipulating push-down lists. The initial set up pushes down lists for storing the input symbol, the replacement symbol, and the name of the list, and also pushes down a temporary storage cell that will hold the current location on the list as the routine progresses. The final cleanup pops up these four push-down lists, leaving them exactly as they were when the routine was entered.

Recursions are not a necessity in list programming, but they are an immense convenience. Conventional iterative procedures could be used, but a great deal of housekeeping (keeping track of all of the lists, sublists and so on) would have to be done explicitly, whereas with recursions the housekeeping is done automatically. Opportunities for recursions abound in artificially intelligent programs. Our bridge program will need a recursive routine for selecting a card to play. The selection will involve looking ahead to the possible plays from the other hands: *i.e.*, what cards will the others then select to play? These selections may be predicted by using the "select a card" routine in the context of the previous selections and the holdings of each other hand. Evaluating the consequences of a selection will also involve a consideration of future tricks; the same "select a card" routine will be used again, so a recursion nat-

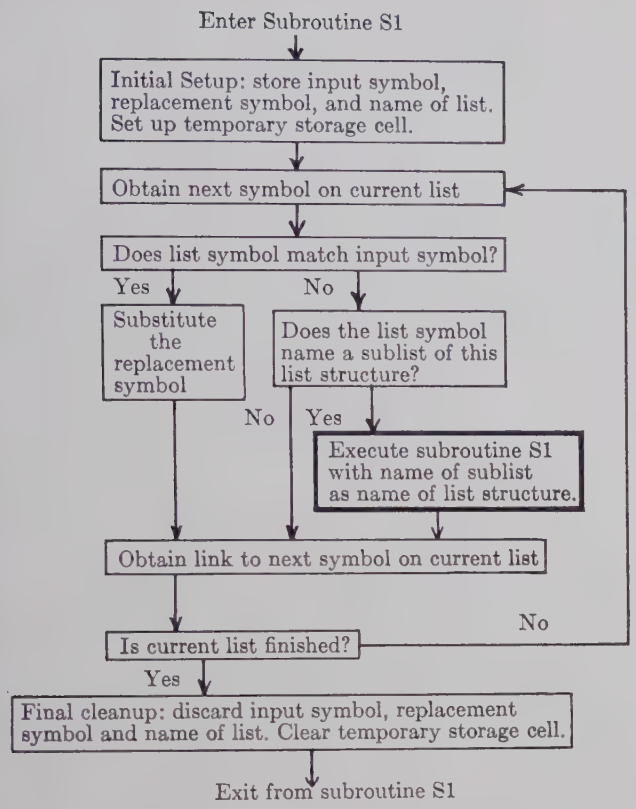


Fig. 3—Flow diagram of subroutine S1, a recursion to substitute one symbol for another on a list structure. The box "Executive subroutine S1 . . . " contains the entire flow diagram of routine S1, including a box "Executive subroutine 1 . . . " that contains the entire flow diagram of routine S1, and so on, in a "baking-powder-can" regression.

urally results. Each successive play that is considered involves a level in the recursion hierarchy of the "select a card" routine. All game-playing programs have such "foresight" recursions. The big problem is when to stop recursing, since the machine seldom has time or capacity to look ahead as far as it might.

Particular Languages

A. IPL

IPL-V, the fifth version of a family of information-processing languages designed by Newell, Shaw, Simon, and their collaborators, is the most widely used information-processing language. It includes over 150 basic processes for handling symbols, lists, and list structures. IPL programs are written as vertical lists of successive instructions, in the same way that machine-language codes are written. In contrast to conventional coding, all symbols used in IPL programs must be in the form of a single letter followed by a relative number. The programmer may not use his own private mnemonic symbols. IPL-V exists as an interpretative routine for the IBM 650, 704, 709, and 7090, as well as for the Univac 1105. The programming manual is published as a paperback by Prentice-Hall, Inc.[8]

[8] A. Newell, Ed., "Information Processing Language—V Manual," Prentice-Hall, Inc., Englewood Heights, N. J.; 1961.

B. LISP

LISP, the list processor designed by McCarthy, Minsky, and their collaborators at M.I.T., is very similar to IPL. It includes some novel technical features designed mainly to improve the speed of operation of the machine. It has fewer basic processes; the programmer must build up his routines from a much smaller set of basic operations. Symbolic mnemonics are allowed, and the programs are written in a horizontal linear notation in the manner of symbolic logic expressions. This notation is oriented toward mathematicians rather than computer programmers. LISP exists both as an interpreter and a compiler for the IBM 704 and 709.[9]

C. FLPL

Gelernter's FORTRAN List Processing Language exists as a set of routines within the FORTRAN system. Specifications for these routines have been published,[6] but the routines themselves are not available, as far as this author knows.

If the programmer needs the resources of an algebraic compiler as well as a list processor, then FLPL would be useful. However, producing the necessary routines from the specifications is not a quick and easy task. An alternative is to imbed algebraically compiled routines within the IPL or LISP structures, since both have the facility for adding processes coded in machine language.

D. Others

Many other programming languages make use of the basic ideas of list memories. Yngve's COMIT,[4] a compiler for handling linguistic material, uses a list memory. Most compilers of algebraic programs use lists to do their compilations. Perlis[10] has described a special type of list called the threaded list for use in compiler routines. Carr[11] has also contributed to the development of general list processing. Fredkin[12] has designed a more elaborate memory structure, called a *Trie* memory, for data retrieval problems.

Uses of List-Processing Languages

The programming languages discussed in this paper have been especially designed for complex, non-numeric problems that have indeterminant storage requirements, that are naturally stated hierarchically and recursively, and that are subject to frequent changes. One type of problem for which the languages are especially suited is game-playing. Of course, game-playing programs can also be written in machine language, but only with great

[9] For further information, write Prof. John McCarthy, Dept. of Elec. Engrg., Mass. Inst. Tech., Cambridge, Mass.

[10] A. J. Perlis and C. Thornton, "Symbol manipulation by threaded lists," *Commun. Assoc. Comput. Mach.*, vol. 3, pp. 195–204; April, 1960.

[11] J. W. Carr, III, "Recursive subscripting compilers and list-type memories," *Commun. Assoc. Comput. Mach.*, vol. 2, pp. 4–6; February, 1959.

[12] E. Fredkin, "Trie memory," *Commun. Assoc. Comput. Mach.*, vol. 3, pp. 490–499; September, 1960.

difficulty and with important limitations. Chess programs written in machine language have generally looked ahead a fixed number of moves, whereas chess programs written in list-processing languages can use recursions to look ahead a varying number of moves, depending on the circumstances. Newell and Simon's chess program,[13] for example, evaluates the consequences of a move by looking ahead move after move until it reaches a position in which there is no possibility of one piece capturing another. The same kind of variable look-ahead could be incorporated into a bridge program, although to our knowledge no such program has ever been written. List languages also provide convenient mechanisms for game-playing programs to use in profiting from past experience. The past experience is often best expressed in terms of sets of associations, the associations among items being represented by their presence on a single list.

Several programs that solve problems or prove theorems have been written in list languages. The first of these was Newell and Simon's logic theory machine,[14] which produced proofs of theorems in logical calculus. Rather than using an all-inclusive algorithm for proving logic theorems, their program used a set of "heuristics," rules of thumb that usually work in producing proofs, but which have no guarantee of success. The heuristics were represented in the program by a list of possibilities; heuristics that worked were moved up on the list, while unsuccessful heuristics were demoted. Following the success of the logic theorem-prover, Gelernter and his associates[15] used the FLPL language to provide a computer with heuristics for proving theorems in plane geometry. More recently, Tonge[16] has written a program in IPL for balancing a complex assembly line in a nearly optimum way, and Slagle[17] has written a program in LISP for doing formal integration.

Problem solving is very similar to game playing, in that a great many possible paths must be examined. The heuristics are intended to keep the program on or near the correct path, and to give it ways of rejecting paths that will lead nowhere. The logic and geometry programs used a means-end analysis, and worked from the solution back to the premises. They sought a subproblem which, if proved, would allow them to prove the required results. In this way, the problem was changed to finding a proof for the subproblem. Newell and Simon, feeling that this analysis is common to a great many problems, have produced a general problem-solving program[18] which uses such an analysis in a general framework.

The heuristic procedures used in the logic and geometry programs are patterned after human behavior. Several attempts have now been made to produce programs that solve problems in the same way that humans solve them, and other programs have been written to simulate other aspects of human behavior. Newell and Simon have adjusted parameters and subroutines in their logic program to simulate the behavior of novices proving theorems in logic. The particular heuristics of a given subject are inferred from a protocol taken of his behavior in an experimental situation, and these rules are given to the computer. The computer program is then a model for the person's behavior.

Computer programs that simulate human behavior have been called information-processing models, to distinguish them from mathematical models that do not postulate specific intervening processes. Feldman[19] has proposed an information-processing model of behavior in a two-choice "probability learning" situation, in which responses are governed by tentative hypotheses concerning the sequence of stimuli, rather than by random events. An information-processing model of human verbal learning has been studied by Feigenbaum,[20] and a similar model has been described by Hovland and Hunt[21] for concept formation. In these models the associations among items are represented by their presence on the same list. Large list structures are used to model complex associations.

List languages are also being used to program linguistic processes. McCarthy is using LISP for programming his "advice taker."[22] This program will accept declarative sentences in English as input, and will abstract the relevant information these sentences and remember it. This information, in the form of relations between objects, is then consulted to answer questions posed in English. A similar approach is being used by Lindsay[23] in an IPL program for language syntax analysis.

List processing has turned out to be a very convenient way of programming "Baseball," a program being developed by the author and his colleagues at the Lincoln Laboratory for answering questions posed in natural English about stored data. The linguistic part of Base-

[13] A. Newell, J. C. Shaw, and H. A. Simon, "Chess playing programs and the problem of complexity," *IBM J. Res. and Dev.*, vol. 2, pp. 320–335; October, 1958.

[14] A. Newell and H. A. Simon, "The logic theory machine," IRE TRANS. ON INFORMATION THEORY, vol. IT-2, pp. 61–79; September, 1956.

[15] H. Gelernter, J. R. Hansen, and D. W. Coveland, "Empirical explorations of the geometry theorem machine," *Proc. Western Joint Computer Conf.*, pp. 143–150; May, 1960.

[16] F. M. Tonge, "Summary of a Heuristic Line Balancing Procedure," The RAND Corp., Santa Monica, Calif., Paper P-1799; 1959.

[17] J. Slagle, personal communication.

[18] A. Newell, J. C. Shaw, and H. A. Simon, "A Variety of Intelligent Learning in a General Problem Solver," The RAND Corp., Santa Monica, Calif., Paper P-1742; July, 1959.

[19] J. Feldman, "Analysis of Predictive Behavior in a Two-Choice Situation," Ph.D. dissertation, Carnegie Inst. Tech., Pittsburgh, Pa.; 1959.

[20] E. A. Feigenbaum, "An Information Processing Theory of Verbal Learning," The RAND Corp., Santa Monica, Calif., Paper P-1817; October, 1959.

[21] C. I. Hovland and E. B. Hunt, "Computer simulation of concept attainment," *Behavioral Sci.*, vol. 5, pp. 265–267; July, 1960.

[22] J. McCarthy, "Programs with common sense," *Proc. Symp. Mechanisation of Thought Processes*, Natl. Phys. Lab., Teddington, England, Her Majesty's Printing Office, London; 1959.

[23] R. Lindsay, personal communication.

ball is best programmed by treating the sentence as a list of words, each word having an attribute list containing its definition in terms of attribute-value pairs. The information retrieval part of Baseball makes use of list structures for storing the data, which are baseball scores, and recursively searches the data for the answers to the question. By using lists for both the definitions of words and the organization of the data, a very general program can be written that is not geared to a specific subject matter. By simply changing the dictionary lists and the data an entirely new context for answering questions can be provided.

The most important advantage of a list-processing language is not in list memories as such, nor in recursions, but in the ease with which larger and larger units of processing can be built up and used as programming units. The programming job can be split into levels so that at any one level very little attention to detail is required. At each level of processing, the programmer can think and write in units that are natural for that level. It is difficult to communicate the profound feeling of fredom and relief that comes from using a list language rather than machine language; the reader is urged to try it for himself.

Computer Synthesis of Character-Recognition Systems*

D. N. FREEMAN†

Summary—**A SAP (Symbolic Assembly Program) package has been developed for the IBM 704 computer to simulate the logical tree of circuitry associated with a character-recognition device. The program has two major inputs: a particular set of logic statements (of AND and OR type) on cards, for flexibility, and tape reels of binary images of ideal or real characters. The output is the "score" of the logic: how many misjudgments it made on the images, what areas of the logic caused the misjudgments, and possible improvements of these areas.**

I. Introduction

TYPICALLY, character-recognition logic circuitry is synthesized under four major constraints: 1) Flexible decision rules are necessary, so the logic can recognize imperfect data patterns; 2) Conversely, rigorous decision rules are necessary, to assure reliable and predictable character recognition; 3) Low cost and minimum circuitry are desirable, for obvious reasons; 4) Development time and development budgets are limited. The fourth constraint, of course, generally supersedes all the others, though if sufficient time, manpower, and money are available, feasibility at least can be proved.

This paper discusses an IBM 704 program, known as SCRIPT (Simulator for a Character Recognition Instrument through Programming Technology), which aids character-recognition logic-design efforts by evaluating a design without the necessity of building an engineering model. This substantially reduces development time and cost and permits investigating many more logic configurations than would otherwise be practical.

SCRIPT uses many conventional techniques, such as simulation and an automatic compiler, plus several new techniques, one of which is the estimation of failure margins. Its primary mechanism is the simulation of proposed logic for a recognition device to which are applied digital representations of characters. The character representations are prepared in binary-bit form before being submitted to the SCRIPT logic model. The performance of the logic is recorded and summarized by the information-retrieval subroutines of SCRIPT. The output thus indicates the ability of the logic tested to recognize the set of character representations supplied to it.

The digital character representations are assumed to be realistic counterparts of legible characters (*i.e.*, identifiable by the human eye and, hopefully, by a mechanical logic), so the performance of the logic is measured by its success in recognizing these characters. If the logic fails to recognize one image in every 10,000, for example, this level of performance can be identified with the logic; logic improvements may improve this ratio to 1:100,000 or better. Thus, the principal assumptions are that the images which the logic processes are realistic, and that the logic can be evaluated constructively by its performance against these images.

* Received by the PGEC, August 18, 1960; revised manuscript received, May 18, 1961.

† General Products Division Development Laboratory, International Business Machines Corp., Endicott, N. Y.

SCRIPT is oriented toward problems peculiar to optical character recognition; however, it also appears to be applicable to such related efforts as magnetic character recognition and speech recognition. SCRIPT offers broad flexibility in its input formats and output options, a consideration only slightly less important to the designers than simulation speed. Thus, other development groups should find little difficulty in converting it to their special needs. Several programming devices have been contrived solely to accelerate the speed of processing.

In addition to SCRIPT itself, this report enumerates other design techniques and 704 methods which have been tested during logic-developjent efforts, plus techniques which have not yet been tested, since their development costs are presently excessive.

II. Structure of the Recognition Device Simulated by SCRIPT

SCRIPT simulates any recognition device which has the following characteristics: It cycles a two- (or three- or one-) dimensional array of bits (a "pattern") through a serial register, with appropriate gating for attempting recognition or passing to the next pattern of adjacent positions, and also for sensing when a pattern has not been recognized within the allowable time interval.

The trees of recognition logic are attached to the serial register in parallel, one tree for each character of the "alphabet." Thus, at any position, the pattern of bits can be recognized as none, one, or several characters. If the pattern is recognized for the first time as exactly one character, a latch is set to identify the pattern to the output of the recognition device. However, the cycling of the pattern through the serial register continues until the cutoff time for recognition, at which time the final validity decision is made.

If, by that time, the pattern has been recognized as no more than one character, it is accepted as a successful recognition. If latches for other characters have been set—either simultaneously at one position or separately —the recognition attempt is unsuccessful, a "conflict." If no latches have been set by cutoff time, the recognition attempt is unsuccessful in another sense; it is a "failure."

The recognition device will send its normal, "recognized" output to one type of document or device (such as punched cards, magnetic tape, or sort hoppers for the scanned document) and all conflict-type and failure-type output to some other document or device (such as a reject hopper for scanned documents or an error punch on a card).

SCRIPT simulates all of these procedures except the scanning and digitalizing of the characters on the documents. However, the output from SCRIPT is designed to retrieve information on the accuracy of the logic, assuming that the patterns have been faithfully represented to the logic. This retrieval is far more elaborate than that of the hardware device; it frees the logic designer from manually analyzing volumes of rejected documents.

(Although SCRIPT has not been designed to catch failures of the scanning device itself, or of the output from the hardware machine, it can indicate a limited class of failures which are manifested as logical errors in a prototype machine: if a pattern fails to be recognized by the prototype, yet is passed successfully by SCRIPT on a rerun through the simulator, the prototype is a far more likely error source than the computer simulation.)

III. Preparation of Character Representations

The images used by SCRIPT are prepared in the following manner: character specimens are placed in a scanning device which scans the printed area with a tiny spot of light. At each position of the spot, logical circuitry assigns the sensor output a digital value (1 or 0) according to such factors as reflectance level and rate of change of reflectance. Each bit (1 or 0) represents an average evaluation over one position of a 48×60 matrix, so that 2880 bits are obtained over a single character. These are again automatically consolidated by a factor of 4 in each direction, using complex data consolidation techniques. The final 12×15 matrix is punched by a scanner-driven punch onto a binary card, to be read through the 704's on-line card reader by SCRIPT. This process is shown in Fig. 1.

Fig. 1.

Since the scanning and data consolidation technique are not part of the SCRIPT project, and since machine circuitry will face the same 12×15 matrix of consolidated information as the simulator in SCRPT, the data collection and reduction techniques will not be treated in this report.

IV. Early Applications of the SCRIPT Program

SCRIPT has been used in two distinct development efforts: exploratory work in the optical recognition of hand-written numerals, and refining and evaluating logic for a machine to optically recognize IBM 407 printing. These have had different goals: the simulated handwriting recognition instrument produced volumes of statistics and patterns for its designers; the printer effort produced sophisticated estimates of the performance of preliminary logic configurations, which suggested improvements to this logic.

For both simulations, however, the underlying philosophy of operation was the same. A rough recognition logic was synthesized by design engineers, using crude frequency tables of the characters in a reduced alphabet (only the ten numerals plus a few special characters).

By various devices, digital character representations were collected and then transcribed onto cards for input to the computer. These samples were stored permanently on magnetic tape by an independent computer program.[1]

During each run of the main program, the logic was simulated in the computer and used to judge these samples. When the logic passed samples properly, that is, when it identified good samples according to their prepunched identification and rejected illegible samples, SCRIPT recorded proper operation of the logic. ("Legibility" of samples was judged by a technician who observed each reduced sample on a video screen and key-punched the appropriate character onto the binary card representing this sample. If the technician felt that the sample was ambiguous or unintelligible, he would indicate this condition by a special punch or by deleting the pattern card for this sample.) For example, SCRIPT was capable of calculating, loosely, the effectiveness of the individual logic tests; a test which always passed when a "legible" sample passed its tree of logic properly (*i.e.*, not as a conflict) was judged a "good" test on the grounds of utility. Likewise, a test which always failed when its tree of logic failed to recognize some other character was adjudged "good" for its inhibiting effect.

When the logic passed or failed samples improperly, SCRIPT recorded a logic malfunction; any test which was failure-causing needed to be altered or deleted. After processing a certain block of samples, SCRIPT stopped the simulation cycling, printed out the summary statistics tabulated, and made whatever statistical decisions the logic designer had specified for the tests and logic.

After processing the block of samples and summarizing the performance of the logic, the computer effort was complete; then, by selecting the failure-causing tests on a statistical basis, altering or replacing them in some reasonable way, and reprocessing the sample, it was possible to determine what improvement (if any) was made. Perhaps the computer could automatically perform some logic modification; however, designer and programmer experience in altering logic is insufficient to encourage automation of this procedure at present.

For the 407-printing effort, SCRIPT has made increasingly elaborate evaluations. A robot recognition device is being used to scan thousands of character representations daily; whenever it misrecognizes a sample, a binary card is punched with the pattern and with the partial identification.[2] The robot is a prototype machine, plus a special output device to prepare punched cards or magnetic tape for SCRIPT. Such a robot is not essential to the usage of SCRIPT, but it enables the logic designer to obtain realistic patterns, particularly those patterns which the present logic, implemented in hardware on the robot, has misrecognized. These patterns are then evaluated by SCRIPT using the same logic or a modification; when sufficient improvements have been found, the logic of the robot is ready to be updated. Thus, the program helps improve the reliability of a partially successful device without the necessity of building a model to test each new development in the logic.

[1] Current investigations include generating images as before, then transcribing them directly onto magnetic tape.

[2] Often there is no way of identifying samples except visually. Since this requires visual inspection and keypunching the computer program can evaluate both unidentified and identified samples.

V. The SCRIPT Simulation Process

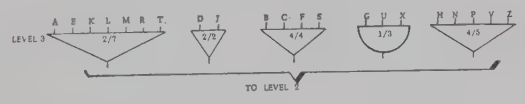
There are six major components of the SCRIPT simulator:

1) Input data, in a matrix array of binary bits (1 = black, 0 = white). (The method of generating these data was explained in Section II.)
2) Trees of logic, one tree corresponding to the logic for recognizing one character.
3) A 180-position shift register.
4) Decision machinery, which interrupts the simulation loop whenever a new pattern is read in or when data are to be gathered.
5) Additional trees of logic, *viz.*, a pretest and alternate tests.
6) Success and failure margin calculations.

A. Logic Trees

Trees of logic are represented to SCRIPT as a deck of Hollerith-punched cards, each card containing the symbolic name of a logical block, its fan-in of other symbolic names, and its logical function. Each tree is an independent logical structure, composed of logic blocks of the m/n type—m inputs are needed for an output signal, out of n inputs available. (The values of m are not restricted to 0, 1, 2, $n-1$, and n, although practical circuitry may dicate the latter values.[3]) The trees of logic are evaluated in parallel in the hardware device, the same binary bit entering the trees for several characters if useful to each of the corresponding logical decisions. A bit to the left side of the logic grid might enter "8" logic as a positive input, but might enter "3" logic as an inverted, inhibiting input. Any desired input may be inverted, as may the output of a block after evaluation.

[3] For example, one logic-development group defines a block as an AND, an OR, an INVERT, or a special logic circuit. This special logic circuit provides an output for either $2/n$ inputs or $(n-1)/n$ inputs, where $n \leq 16$. Thus, if the inputs to level 3 (which come from level 4) are labeled A, B, C, D, $\cdots$, then a part of level 3 might appear as:



where triangles indicate AND or special circuits, and the semicircle indicates OR circuits.

The blocks are arranged into levels so that the highest-level blocks have all their inputs from the pattern matrix, those of the next level have inputs from the highest level (and possibly the pattern matrix), and so on. The decision level is level one, where each block determines the success or failure of its entire logical tree. Up to nine levels of logic are provided with the present program, an adequate maximum thus far.

The logic designer can arrange inputs from the pattern matrix and from higher-level blocks into any desired configuration to feed lower-level blocks, corresponding to the degree of complexity in decision-making. He might, for example, specify that 100 of the 180 available inputs be logically treated by level-9 logic; level-9 outputs, and, say 60 of the remaining 80 inputs might be treated by level-8 logic; and 10 of the last 20 inputs might be combined with level-8 and level-9 inputs by the level-7 logic. The last 10 potential inputs might be ignored as ineffectual for identifying any character.

The flexibility of the foregoing structure is amplified by "zone logic," a logical structure to locate the characteristics common to several characters simultaneously. For example, several bits from the pattern matrix might be logically examined for arc (or absence of arc), crossing, rectilinearity, and so on. The zone logic would present outputs to the entire alphabet of characters at once, effecting a large saving in cost of components in the hardware device. In SCRIPT, nine levels of zone logic are available, making 18 levels of logic available in all between the pattern matrix and the decision level for each tree.

B. Shift Register

The shift register is a realistic simulation of the movement of a scanning device. Schematically, the recognition logic looks as shown in Fig. 2. The physical scanning device searches vertical columns of the image field, sending information to a buffer. After one column has been scanned, the scanner moves one field to the left in a motion as shown in Fig. 3.

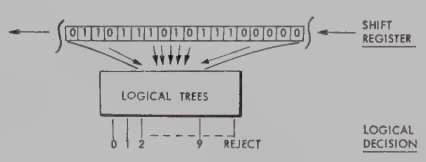


Fig. 2.

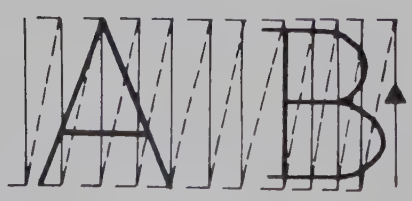


Fig. 3.

Thus, if the cells of the image are designated

A1 B1 · · · M1
A2 B2
. . . . . .
A15 B15 · · · M15,

three consecutive contents of the shift register might be

0	M15	M14		M1	L15		A2

M15	M14		A1

M14	M13		A1	0

As the bits move to the left through the 180-position shift register, they are sampled by the inputs to the trees of logic. For example, one input to a certain logic block might come from the 180th position of the register, and would thus contain bits A2 and A1 of the image at the first two of the above three sample times. Thereafter, the position would contain 0 (*i.e.*, white) until the next character in the line of printing appeared under the scanner.

C. Decision Machinery

If, in a manufactured character-recognition device, a sample does not set at least one "success" latch (*i.e.*, pass at least one logic tree) during the shifting, it is a failure; the recognition device then rejects the paper document or takes some similar action. SCRIPT has the ability to record "near passes," and the pattern can be reprocessed to extract substantial information about what caused its failure. (Of course, if the pattern is sufficiently illegible, recognition is not desirable, but we are assuming that all patterns used in logic development are "reasonable.")

If the sample passes the logic for exactly one character, then it is a success. Ordinarily, this is of little statistical interest except as a simple numerical count. However, information on specific logic configurations particularly useful to the decision process is valuable for circuit minimization and for strengthening the success margin. If a given logic block never actively contributes to the decision-making process and performs so inhibiting function to prevent an incorrect process, the block should be altered or omitted. Elsewhere, another block might be strengthened by a slight alteration, so that it would pass more frequently when the entire pattern was successful; this too would represent a strengthening of the logic. If a sample is identified by the simulated logic as more than one character, it is a conflict. In addition, there is the possibility of single substitu-

tions if the samples are identified to SCRIPT.[4]

Single substitutions must be avoided in every reasonable way on a production machine, since they represent grave output errors; detection is mandatory, even if more failing images must be incurred as the penalty for screening out potential substitutions. For either substitutions or conflicts, SCRIPT is capable of counting test performances and other data on the offending logics.

D. Pre-test and Alternate Tests

The pre-test is a computational device with no circuitry analog. It has saved over 80 per cent of the potential simulation time by eliminating those positions of the pattern which could never pass any logic. The pre-test is a small tree of logic which looks for "central positions" of the pattern in the shift register. Essentially, it looks for a predominantly dark central area with white borders. If a pattern is not "centered" in the shift register, the logical trees will expect black bits where there are none and failure will invariably result at these positions.

For the simulation of a 407-printing recognition device, 60 shift positions were specified by the logic designer. The pre-test permitted logical-tree evaluations at only about six of these positions. Since the pre-test was barely one per cent of the size of the main logic, the simulation time was largely restricted to potential recognitions.

The alternate tests are an option to indicate where the logic being simulated could best be improved. For any test of the logic being simulated, several alternate tests can be supplied. Essentially, they are dead-end tests which have no circuitry analog and no decision function in SCRIPT; they use more or less the same inputs as the main logic test (see Fig. 4).

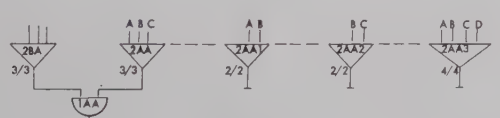


Fig. 4.

Tests 2AA1, 2AA2, and 2AA3 are alternates to test 2AA. Whenever the logical tree using 2AA passes or nearly passes some pattern, the alternate tests will be evaluated. Since each logical tree rarely passes a pattern, considering the number of characters in the alphabet and the number of positions at which each sample is tested, alternate-test evaluation consumes a negligible amount of the simulation time.

If, for example, test 2AA2 passes (*i.e.*, yields an output) more often than do 2AA, 2AA1, or 2AA3 when the pattern is correctly recognized, and if it does not pass patterns which should be screened out, it is clearly a superior choice. The logic designer can introduce it into the main flow of a subsequent modified logic.

E. Success- and Failure-Margin Calculations

A rough margin of failure or success is calculated for each pattern against each logic at each shift position where the pre-test was passed. If this margin is significantly small, additional data-processing techniques can be used on the pattern.

For example, if a pattern never passes any logic during the shifting, it is a failure. However, since these test patterns are assumed to be legible, it is desirable to retrieve information about the causes of failure. SCRIPT implements the following assumption: if a pattern is "close" to passing a logic properly, and if the shift positions where it came close are known, then the performance of the individual tests at these positions is significant. Those tests which contributed to the failure of the logical tree should be relaxed or replaced.

For failing patterns, then, the margin of failure is significant; whenever it is "low," information should be gathered. However, a pattern is not known to be a failure until the shifting is completed. Thus, the program must either completely rerun a failing pattern in some information-gathering mode or it must save sufficient data during its initial processing of the pattern to permit a restricted rerun, returning to gather test data at the crucial positions only.

SCRIPT uses the latter strategy, summing the individual-test failure margins as the margin for an entire logic. Thus, at each shift position, 13 such margin-sums are calculated for the 13-character alphabet. If any are significantly low, the character and shift positions are recorded for later processing. In detail, the failure margins are calculated as follows:

1) Certain tests have been identified as being important to the decision. (In the 407 effort, all tests below the third level were used.)
2) At the time that these tests are evaluated against a pattern, the margins of failure are calculated and accumulated. For example, if some test requires three inputs to pass, but only receives one input, a failure margin of two is recorded. If the same test receives three or more inputs—and thus passes—no failure margin is recorded. Unimportant tests [by the designation in 1)] do not enter the sum.
3) If the pattern passes some logic, then its failure-margin sum is meaningless.

During the effort to polish 407 logic, a failure-margin limit of one or two provided satisfactory information on failing patterns; only a few patterns never came within two "important" inputs of passing the proper logic.

[4] An example of a "single substitution" would be an image which is visually identifiable as a "3," but only passed "5" logic.

This limit is a parameter specified to SCRIPT by the design engineer; it represents his estimate of how many important tests may fail, and by how much, for the pattern to have "barely missed." Since the trees of logic may vary considerably between characters of the alphabet, a vector of these failure-margin limits is specified by the design engineer, one limit corresponding to each tree. The determination of the limits is left to the design engineer, since it is he who identifies the crucial tests from which the failure margins are calculated.

Another strategy—used successfully on the handwriting effort—was to let the computer pick those shift positions of a failing pattern for which the failure margin was smallest, without specifying an upper bound. In this case, the designer specified that only the best five positions, for example, were to be re-examined.

Margins of success are calculated for successful patterns in similar ways. The objectives of success-margin calculations are to eliminate misrecognitions on the one hand and to strengthen correct recognitions on the other hand. The margin of success for a pattern is the sum of the excesses of successful crucial tests. (These tests are the same designer-identified tests used for failure-margin calculations.) Thus, if a crucial test failed or passed exactly (*e.g.*, for a normal AND block), no margin would be accumulated; if a 3-of-4 test received four positive inputs, a margin of one would be recorded, and so on.

To conserve processing speed, the compiled sequence of logic is re-entered only at the SCRIPT command where evaluation of the crucial tests commences. That is, the higher-level test results at this shift position are still in their counters [see 1) above] and the logic needs to be rerun in the success-margin mode only from the third level down. Since failure is more common than success (only one logical tree of 13 is likely to pass, at most) the patterns are evaluated first in the failure-margin mode. If they pass, they are then briefly re-evaluated in the success-margin mode. (Both margins could be evaluated simultaneously, but the computing time would be considerably lengthened and the success-margin sums would be valueless for the failures—over 90 per cent of the evaluations.)

The same limits are used for identifying nearly-failing successes and nearly-passing failures, although this is not a necessary tactic. The ultimate objective of all the "near" calculations is to sharpen the focus of the logic on characters which are neither strong successes nor strong failures. When a good logic system is being refined, the incidence of failure or misrecognition is extremely low for normal patterns, and this technique may prove of extraordinary value; it can seek out potential failures and conflicts and adjust the logic accordingly.

The typical output, shown in Table I, illustrates the function of the decision machinery and the margin calculations. It permits the logic designer to study individual patterns rapidly and effectively.

TABLE I

Conflict Identification	Character 3	Writer *A*	Type Font *u*
Serial No. 12345			
Shift Position	2 6 6	8 9	
Characters Passed	3 5 3	5	
Margin of Success	1 1 2	0	
Near Conflicts	5	3 3	
Margin of Failure	2	1 1	

The first two lines identify the pattern, and show it was a "conflict." The block of data which follows describes its performance against all logic systems to which it was applied.[5] The third line enumerates those shift positions at which the image passed or came crucially close to passing the logic for various characters. The fourth and fifth lines indicate which logic (if any) the image passed, and by what margins. The last two lines indicate near passes (conflicts) and the failure margins. Thus, this pattern passed as "3" by a margin of one and failed "5" logic by a margin of two at shift-position 2, passed both "5" and "3" logic at position 6, and so forth.

VI. Details of the IBM 704 Script Program

The computer program divides into five phases:

1. Input (header cards and logic cards).
2. Compilation of the logic.
3. Initialization, including set-up of the processing and output options.
4. Simulation loop, plus per-character output and data collection.
5. Summaries.

A. Input

In the first phase, cards of six different types are read in:[6]

1) Header cards, specifying the dimensions of the data grid, the alphabet being used, the date, the options of pre-test and alternate-test usage, and the output options to be used.
2) Cards specifying the logical trees. These cards contain the following information about each block: number of inputs; number of energized inputs required for success; level of the block within its tree of logic; whether the output is to be inverted; and the list of inputs—both from the pattern matrix or from other blocks including indicators for inversion if necessary. During the compilation of logic, SCRIPT automatically checks the internal coherence of the logic by searching for the definition of each input; it must be either another

[5] This block format can be repeated if the volume of data exceeds one horizontal band of printing.

[6] These cards are placed on magnetic tape off-line and read in from tape to conserve processing speed.

higher-level test or a bit from the pattern matrix.

3) Similar cards for the pre-test.
4) Similar cards for alternate tests. If desirable, small trees of logic can be constructed to feed these alternate tests with the whole apparatus contained in the alternate-test framework. Thus, alternate tests can have inputs from three sources: the pattern matrix, the main logic, and from other, higher-level, alternate tests. Comparisons of alternate and main-logic tests are made only for appropriately-identified alternate tests.
5) The shift register movement being simulated, in terms of its alphameric coordinates. Fig. 5 displays the correspondence between pattern and logic. During the shifting, it is actually the logical search that moves; the pattern is unpacked into a static buffer. This accomplishes the same movement as if the pattern moved since only the relative positioning is important.
6) The limiting margins for near-success and near-failure (see Section IV-E).

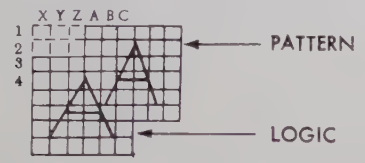


Fig. 5—The logic is positioned at position X3 with respect to the pattern.

B. Logic Compilation

Each tree of logic is compiled into an efficient sequence of 704 instructions. These sequences are entered by the 704's usual linkage direction (TSX exit, return by tagged transfer). The sequences are stacked in cores, one immediately following the other. Thus, the only information needed to enter a sequence is its first address. The list of addresses is accumulated during compilation: the pre-test first address, the first address of zone logic, the first address of 0-logic, 1-logic, etc., and of the alternate-test logic associated with each tree.

The compilation of logic is a one-time affair for each computer approach (it can be saved by dumping on tape, to be reloaded if needed for subsequent runs; for example, the SHARE 704 routine ELSAV1 might be used). For the 400 logic blocks of the 407-printing study, compilation time has been less than three minutes.

The basic module of 704 instructions to test one block is as follows:

1) CLA (Call up the contents of a cell, this value counting the number of inverted inputs to the block.)
2) ADD (Add in the binary value of the first uninverted input.)

ADD (Add in the binary value of the second uninverted input.)

. . .

3) SUB (Subtract the value of the first inverted input, if any.)

. . .

4) SUB (Subtract the contents of a cell, the contents being the numerator of m/n.)
5) STQ (Set the success indication for the test; the test itself will follow and, if it fails, the output indication will be reset to failure.)
6) TPL *+2 (The accumulation of inputs and testing has been in the decrement field; the test is successful if the accumulator is non-negative, and the counter is then left set by 5). The "set" value remains available in the multiplier-quotient register during the entire simulation.)
7) STZ (Reset the success indicator to zero—failure—if the accumulated value is negative.)

In 1), the accumulator is reset to the number of inverted inputs to the block (possibly zero). In the instructions 2), the inputs are brought in, one by one. They are either PZE or PZE, , 1, according to whether the preceding (higher-level) test failed or passed, or according to the "color" (white or black) of an input from the matrix.

If the inputs are inverted, the combination of instructions 1) and 3) will bring in the proper value; *i.e.*, for each inverted input, either $1-0=1$ or $1-1=0$, where the two numbers on the left side of either equation represent the values from instructions 1) and 3), respectively.

Each bit of the pattern itself is unpacked into a single 704 word: 1 or 0 in the decrement field corresponding to black or white, respectively. The unpacked pattern will be termed "in the home position."

Now, the pattern-matrix inputs correspond to instructions 2) of the following form:

```
ADD     INPUT1, C
ADD     INPUT2, C
```

where INPUT1 and INPUT2 are the absolute addresses of the unpacked bits. The pattern has been unpacked by columns into a sequence of consecutive core-storage locations; on either side of this block of storage, "white" borders of some 100 locations have been cleared to zero. Thus, the descending storage locations would contain the following quantities: 0, · · · , 0, (bit A1), (bit A2), · · · , (bit An), (bit B1), (bit B2), · · · , (bit Nn), 0, 0, · · · , 0, where "n" represents the vertical dimension and "N" the right-most alphabet coordinate.

If the pattern is to be tested by the logic at the home position (*i.e.*, position A1), index-register C is pre-set to 0. Thus, each input will be addressed at its unpacked

location. If the pattern is to be tested at position A2, index register C is set to 1. Assume that the vertical dimension is 15. Then, to test the pattern at B1, index register C is set to 15; for testing at position B3, index register C is 17, etc. As shown in Fig. 5, index register C would be $(-3)*(15)+2=-43$.

For inverted inputs, the addressing is exactly the same. If the test is at level one, the decision level, instruction 6) is modified and an instruction 8) added:

```
6) TPL   *+3
7) STZ        (Reset the test indication as usual.)
8) SLN     2  (Turn on a sense indicator, which
               can be tested by the main program
               when control is returned from the
               simulator. If the indicator is on,
               non-recognition has occurred for
               this logical tree.)
```

For those crucial tests whose failure or success margins can enter the accumulated margin, instructions (9)–(15) are added:

```
9) SLT     3  (If another indicator has been pre-
               set by the main program, the margin
               of success is to be calculated, and
               the testing of the indicator causes a
               skip to 11).)
10) TRA  *+5  (Otherwise, transfer to another
               branch of instructions to accumu-
               late the failure margin.)
11) SLN    3  (Restore the success-mode indicator,
               which has been reset by testing in
               9).)
12) TMI  *+4  (The accumulator still contains the
               signed margin of overfulfillment or
               underfulfillment. Exit if negative.)
13) STD  *+1  (Since the success mode used this
               branch, the test was over-fulfilled.
               Add the margin to index-register
               B.)
14) TXI  *+2, B, ,
15) TMI  *-2  (From instruction 10), the failure-
               mode branch involves a test.
```

The complete logic for one tree would appear as follows:

```
CLA—      ENTRY POINT
          A highest-level test for this logic.
STZ—
CLA—      Another highest-level test, or a lower-level
          test.
STZ—
 . . .    Other tests, decreasing by level.
STZ—
SLN 2     The decision is made.
TRA 1, A  Return to the "driver" program, with
          index-register B containing the failure or
          success margin.
```

The instructions to enter this sequence are:

```
LXA SHIFT, C  (Set index register C to the shift-
               position value.)
CLA ADDRESS   (Place the entry-point address into
               the TSX instruction. Ordinarily,
               this is itself an indexed quantity.)
STA      *+3
LXA ZERO, B   (Zero the margin accumulator.)
LDQ 1INDEC    (Supply the counter setter.)
TSX      , A  (Exit to compiled logic.)
SLT      2
TRA SUCCESS   (If a success.)
(Failure)     (If a failure.)
```

In addition, sense light 3 must also be set for the desired mode in advance of the TSX command.

C. Initialization

After the logic has been compiled, the header-card data is used to set gates and subroutines for the collection of information during simulation. Of the large number of output options available, the logic designer will ordinarily use only a few, aimed at the aspects of his logic which need careful study. The gates and subroutines are self-setting and are triggered by 30 one-character fields on the header card.

A this point, just before simultation begins, two options are available: pattern skipping on the input tape (if a former run is to be completed), and dumping the entire core-storage record onto a self-loading tape (in case of subsequent machine failure or other interuption to the run). At any point during processing, throwing a console switch will cause the simulator to stop cycling patterns, skip over the data tape to its end of file, and again make available the two options of record-skipping and machine-dumping. Alternatively, a third option—to terminate processing and summarize the statistics—can be keyed in by another switch.

D. Simulation

The simulation loop is a small subprogram that reads in patterns, unpacks them, enters the various testing sequences, and tests the status of each return to determine what other testing and data processing are needed. The flow chart of the simulator is shown in Fig. 6; the output data and summaries are described in Section VII.

Using the 400-block logic of the 407-printing project, SCRIPT cycled one pattern through 60 shift positions, evaluated the pre-test at all of these positions, evaluated the 13 trees of logic at the positions passing the pre-test (three to ten positions, typically), and prepared the output data for a BCD tape—all in one second. The writing of the output tape varied, according to the volume of statistics and especially to the option of the per-pattern output, from 0–7 seconds per-pattern.

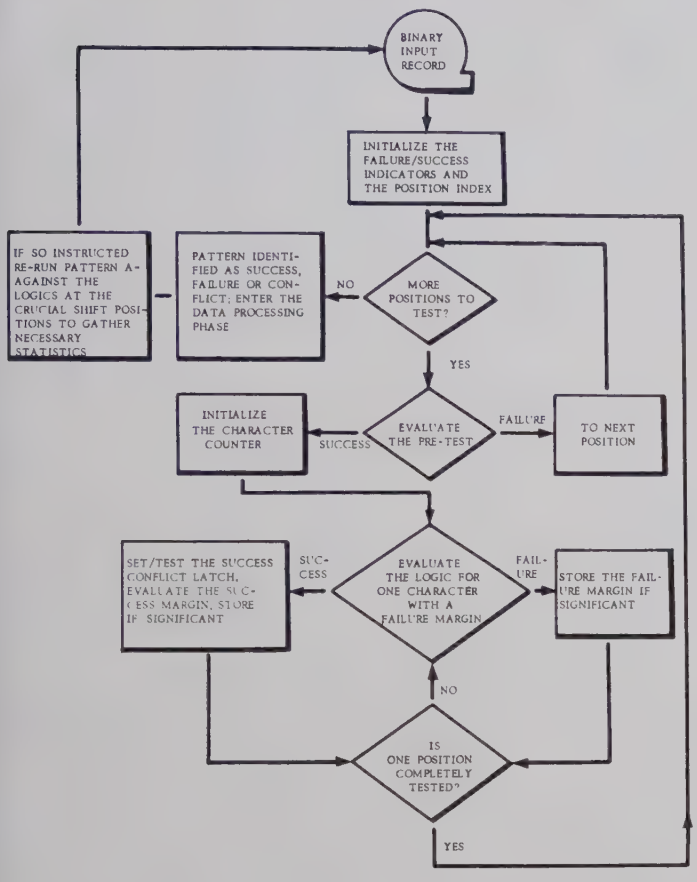


Fig. 6.

VII. Details of the Sample Data; Other Possibilities

It is important that the samples used in this logic-development procedure be realistic. Three other aspects of the choice of samples are also significant to logic development: 1) Should new data be continually tried against an evolving logic? 2) If not, should the entire file of retained data be tried against each modification of the logic? 3) Can random samples of the retained file accomplish satisfactory testing? Or can misrecognized samples from an earlier computer pass? Or can realistic samples be constructed from earlier patterns by random alterations or by the use of frequency-table sampling? The problem of generating realistic samples was solved by transcribing the scanner output onto punched cards. In other words, paper documents with sample printing or handwriting were placed under the scanner, and the output was punched directly into a binary card.

However, this technique was not entirely satisfactory for polishing the already-workable logic for 407 printing. The percentage of misrecognized samples dropped to such a low figure that the computer simulation was wasted in correctly passing satisfactory samples. Thus, the prototype recognition device, implementing an early version of the logic, was an important tool for refining the logic. This robot was designed for punched-card output, the cards showing misrecognized patterns in reduced (12×15) form, and furnishing a good supply of difficult, critical patterns.

The limited robot output might not be entirely satisfactory, since it could bias the logic toward the difficult samples which the robot logic misrecognized. Thus, the logic might be altered to the extent that it would misrecognize samples that originally had passed. Fortunately, the use of difficult samples has not perceptibly biased the logic to date.

The cost of generating new data is important to the development strategy. At the start, the 407 effort relied on 1000–2000 samples of the 13-character alphabet, using a subset of these samples for each run. Using the robot, new data with high diagnostic value have become available; the present strategy is to take several thousand rejects per day from the robot and evaluate them with the 704 image of the logic to find where improvements should be made.

Other logic-development efforts have stored larger or smaller initial files of data; the larger the file, the better the reliability of the logic, but also the higher the expense of gathering and processing the data.

It is feasible to generate all data as needed, perhaps feeding them to a real-time simulation on the computer. As a permanent strategy, however, random sampling of a large file would seem somewhat more satisfactory; this would depend, of course, on the speed of computer access to the large file. Another strategy might be the automatic generation of new patterns by the computer; a variant of this technique is constructing frequency tables from the original patterns, thereby obtaining a statistical model of each character of the alphabet.

For any future logic-development effort which begins without the benefit of realistic samples—without a scanner, for example—the use of ideal characters is recommended. Beginning with a reasonable representation of a visual image, one can convert this picture to a pattern of bits and try one or more of the above techniques, such as controlled aberration of the pattern, to multiply the number of samples.

VIII. Details of the Output

The output of SCRIPT fulfills two broad objectives: to demonstrate the ability of a proposed logical system to identify realistic character patterns, and to suggest, by statistics computed from these patterns, where improvements in the logic might best be made.

The output options of the present program represent techniques which have survived these tests of utility; several other techniques have been discarded as being less useful, still other techniques are likely to be useful but are too expensive for present computers. All three categories will be discussed in this section.

SCRIPT delivers outputs in one of two primary modes:

1) Identified patterns (the tape record specifies the character).
2) Unidentified patterns (the computer must try to identify the pattern as a character).

The file of patterns is then divided into three categories by the logic:

1) Failures
2) Successes
3) Conflicts (recognition as more than one character; in the identified-patterns mode, single substitutions can be detected, but they are classified as "successes" in the unidentified-patterns mode.)

Each category is further divisible according to which character each pattern is (or seems to be). Each set of individual-character results can be subdivided by the near-failures (if any) of characters that passed and by the near-successes (if any) of characters that failed. With SCRIPT, the results of individual tests can be collected and displayed according to any desirable subcategory.

The entire mass of data for one pattern is available after cycling and testing it against the simulated logic. During the initial shifting a record is kept of the crucial shift positions: where characters passed, by how much, and where and how they almost passed. After the pattern is identified as a failure, success, or conflict, a brief return can be made to the crucial shift positions and trees of logic if data is to be gathered on the individual tests or their alternates.

The heavier volume of data resulting from the individual-test results requires intermediate storage during the processing of a large file of patterns. This data is dumped onto a binary tape, each record consisting of a sequence of 0–1 indicators—representing test failures and successes—plus a label to indicate which characters have been passed or nearly passed. During the summary calculations, when the compiled logic is no longer needed, most of core storage can be used to buffer in these records and to accumulate statistics.

To date, both the handwriting and 407 projects have concentrated on analysis of the failure and conflict statistics, plus the individual-test performances at the crucial positions. As the failure and conflict incidences drop to tolerable levels, the logic designers expect to rely more heavily on the weak-area analyses by the "margin calculations" and on the suggestive comparisons generated by the alternate-test statistics.

With the alternate tests, SCRIPT is capable of preparing outputs such as is shown in Table II.

TABLE II

Summary of Failures of the Character 3 at Crucial Positions

Main Logic Test Name	Alternate Test Name	Frequency: Main Passed, Alternate Failed	Frequency: Both Passed Or Both Failed	Frequency: Main Failed, Alternate Passed
2AA	2AA1	30	50	2
	2AA2	15	52	15
	2AA3	0	28	54
2AB	2AB1	0	82	0
	2AB2	1	24	57
	2AB3	3	9	70

For this example, the obvious choice of an alternate to test 2AA is test 2AA3. It is at least as good as 2AA at all crucial positions, and it is better at a substantial number of positions. (The designer might want to be assured that the substitution of 2AA3 for 2AA would cause no conflicts; for this, he would consult the appropriate comparison for successful "1"s, "2"s, "4"s, etc.)

The decision for test 2AB is not quite so simple. Alternate test 2AB1 had an identical performance with test 2AB, and it is valueless for the improvement of failing "3"s (although it might be better or worse for other test-improvement needs). Test 2AB2 was worse than 2AB at only one position, and it was substantially better at 57 positions. Test 2AB3 was even better, if the three additional failures can be tolerated, for it gained 70 successes over the original test and 13 over the next best test.

The 704 can be programmed to search these numerical summaries, select the best alternate for each test, then—if SCRIPT finds an alternate which is superior to the original test—recompile the trees of logic and reprocess the patterns. A slight variation to save 704 time would be to run only those patterns which failed (or conflicted) against the appropriate logics, temporarily ignoring the possibility of additional conflicts or failures being induced by the logic changes.

In addition to test statistics and summaries of the performances of the trees of logic, SCRIPT can print out pictures of the patterns in any category (*e.g.*, all failing "3"s). These patterns are furnished with horizontal and vertical guides so that the logic designer can study specific patterns. [(See Fig. 7(e).]

IX. Autogeneration Possibilities

The preceding section presented the suggestion that SCRIPT select the best alternate tests to replace inferior tests. The entire procedure of test generation and alteration can be automated, although the cost for a 704 effort of this type appears unreasonable with present techniques.

In order of feasibility and utility, the following steps could be taken:

1) Automate the generation of alternate tests.
2) Automate the generation of frequency tables from the "best" positions of patterns; these tables would be used for high-level test correction.
3) Use these frequency tables in a "symmetric-difference" technique to redesign the entire logic.

	CHARACTER ID	PRE-SHIFT	TYPE FONT	SAMPLE NO	UPPER/ LOWER CASE
1 SUCCESS IDENTIFICATION	3	4	01	91108	U

SHIFT POSITION	25 B-2	31 C-2
CHARACTERS PASSED		3
MARGIN OF SUCCESS		0
NEAR CONFLICTS	3	
MARGIN OF FAILURE	1	

SHIFT POSITION		31 C-2								
CHARACTER-3	0	1	2	3	4	5	6	7	8	9
LEVEL 1	1									
LEVEL 2	1	1	1	1	(B) 1	1	(C) 1	1	1	1
LEVEL 3		1	(A) 0	1	1	1	1	1		
LEVEL 4		1	1							

a

	CHARACTER ID	PRE-SHIFT	TYPE FONT	SAMPLE NO	UPPER/ LOWER CASE
2 CONFLICT IDENTIFICATION	(D) 4	1	01	91108	U

SHIFT POSITION	18 B-3	32 D-3	24 C-3
CHARACTERS PASSED	¤	¤	4 ¤
MARGIN OF SUCCESS	0	0	0 0

SHIFT POSITION		24 C-3								
CHARACTER-¤	0	1	2	3	4	5	6	7	8	9
LEVEL 1	1									
LEVEL 2		1	1	1	1	1	1	1	1	1
LEVEL 3		1	1	1		1	1		1	1
CHARACTER-4	0	1	2	3	4	5	6	7	8	9
LEVEL 1	1									
LEVEL 2	1	1	1	1	1	1	1	1	1	1

SHIFT POSITION		32 D-3								
CHARACTER-¤	0	1	2	3	4	5	6	7	8	9
LEVEL 1	1									
LEVEL 2		1	1	1	1	1	1	1	1	1
LEVEL 3		1	1	1		1	1		1	0

SHIFT POSITION		18 B-3								
CHARACTER-¤	0	1	2	3	4	5	6	7	8	9
LEVEL 1	1									
LEVEL 2		1	1	1	1	1	1	1	1	1
LEVEL 3		0	1	1		1	1		1	1

b

	CHARACTER ID	PRE-SHIFT	TYPE FONT	SAMPLE NO	UPPER/ LOWER CASE
3 IDENTIFICATION OF A COMPLETE FAILURE	(E) 4	2	01	91108	U

THERE WERE NO CLOSE CHARACTERS.

c

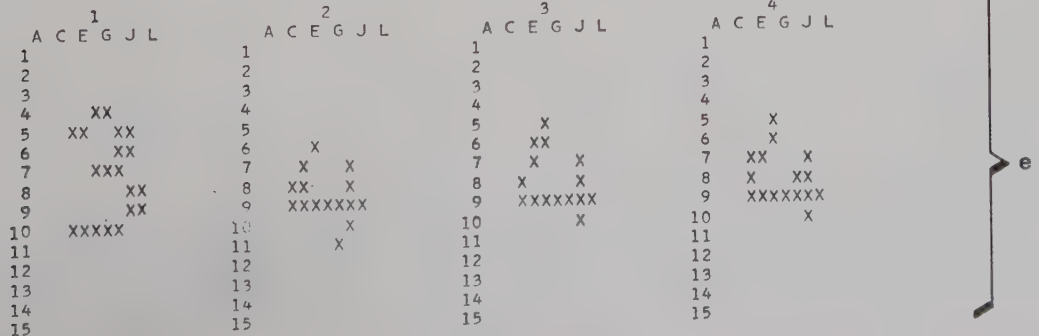
	CHARACTER ID	PRE-SHIFT	TYPE FONT	SAMPLE NO	UPPER/ LOWER CASE
4 IDENTIFICATION OF A COMPLETE FAILURE	4	3	01	91108	U

SHIFT POSITION	25 B-2	31 C-2
NEAR PASSES	¤	¤
MARGIN OF FAILURE	1	1

SHIFT POSITION		25 B-2								
CHARACTER-¤	0	1	2	3	4	5	6	7	8	9
LEVEL 1	0									
LEVEL 2		1	0	1	1	1	1	1	1	1
LEVEL 3		0	1	1		0	0		1	1

SHIFT POSITION		31 C-2								
CHARACTER-¤	0	1	2	3	4	5	6	7	8	9
LEVEL 1	0									
LEVEL 2		1	0	1	1	1	1	1	1	1
LEVEL 3		1	1	0		0	1		1	0

d



e

Fig. 7.

4) Begin with no logic at all (but, necessarily, with identified patterns). Center the patterns in a matrix by some "bit center-of-mass" technique to compile frequency tables for each character. Create a rough logic, then proceed as in 2) and 3).

The generation of alternate tests would depend on the collection of individual-test summaries for the crucial positions. Troublesome tests might be altered in three ways: by raising or lowering the requirements on the inputs, by deleting inputs altogether, or by bringing in new inputs. For the first, it would be relatively easy to instruct SCRIPT to change the test-requirements constants; if more successes were desirable, an m/n test might be changed to $(m-1)/n$ or $(m-2)/n$, and similarly for more failures of an inhibiting test.

To implement step 2) above, frequency tables for each character might be accumulated during the second pass against the logic (after the first pass has identified the patterns as failures and saved the crucial-position identifications). From these tables, high-frequency bits might be added to the blocks which they border, and low-frequency bits might be deleted from these blocks.

Step 3) above is an even more sophisticated procedure. After the frequency tables have been accumulated, SCRIPT could use them to entirely recreate the high-level blocks. For example, suppose that frequency tables have been compiled for all "3"s—both successes and failures—at their best shift positions. Then, adjacent high-frequency bits can be joined in the two-bit, three-bit, or four-bit synthesized tests which are most successful.

Step 4) is the most formidable technique that we have considered. The only inputs to the computer program would be identified patterns, plus the complex set of decision rules for synthesizing and altering logics. Each pattern would be read into the shift-register, centered by some reasonable technique, then added bit by bit into its character's frequency counter. These frequency counters would be used to generate a highest-level group of tests, all of whose inputs would be bits. Then, the patterns would be rerun against this loose "logic," to determine the crucial positions of the patterns. (In fact, these crucial positions would represent a rough centering of the patterns against the logic.) Frequency tables could again be accumulated on the bits of the pattern at these crucial positions, and the entire procedure could be iterated.

When satisfactory highest-level tests have been created—and the frequency tables sharpened to define each character clearly—the techniques of steps 3) and preceding could be used to synthesize and correct the lower-level logic.

Throughout such a highly automated procedure, human monitoring would be necessary to be assured that the synthesized logic did not take an unreasonable form and also to determine when each synthetic or corrective procedure had reached a reasonable logic and significant gains were no longer likely.

Appendix

Sample Outputs

A sample output from SCRIPT will show the great detail to which the simulation results can be tabulated. This display is unusually elaborate; ordinarily, the logic designer will be interested in only some subclass of this output. In addition, the logic designer can monitor the output during a computer run with SCRIPT, either by instructing the 704 operators to accumulate only a certain number of pages of elaborate output per file of samples or by personally observing the output as it is calculated and throwing the appropriate console switches.

The first portion of the sample output, Fig. 7(a), displays the performance of a successful character "3." The first line below the column headings describes the final computer judgment: a success. Following this comes the summary of the character's performance: it came within a margin of 1 of passing 3-logic at shift-position 25, and it passed 3-logic with no margin of success at shift-position 31. The complete identification of the tape sample is unpacked and displayed here, *i.e.*, pre-shift, type of font, etc. Note that this character passed the proper logic only once—by the narrowest of margins—and thus seems "close to failure" in some empirical sense. The remainder of the block [Fig. 7(a)] is in the form of a matrix array of three values: 0, 1, and (blank); these mean respectively that the indicated tests failed, passed, or were not used in the logical tree. The level index runs vertically, the middle character of the test name runs horizontally in ten columns (0–9), and the last character of each test name indicates the logical tree to which the test belongs. Thus, the matrix entry labelled Ⓐ shows that at shift position 31 (C-2, relative to the home position of the logic, as defined earlier), test 323 failed. Entries Ⓑ and Ⓒ show that tests 243 and 263 passed; the blanks indicate that there were two tests at the fourth level, seven at the third level, and so on.

The second section of sample ourtput, Fig. 7(b), records a conflicting character whose true identification is "4," as indicated at Ⓓ. The dump data shows that the character passed □-logic and 4-logic at shift position 24. □-logic tests were dumped up to the third level, 4-logic only up to the second; the dumping levels are not controlled by the data, however, but by the header-card punching for each character.

As the dumps and summaries indicate, the character passed the conflicting □-logic three times, the correct logic only once.

Fig. 7(c) is a short diagnostic for the next character. Since the character never came within the prescribed margin of near-success for any logic during the shifting,

SCRIPT printed this information and took no further action, *i.e.*, no statistics were gathered. The character was identified on tape as a "4," as shown in Ⓔ.

Another failing "4" is tabulated in Fig. 7(d). However, twice it came within one input of passing "□"-logic—an undesirable situation—and never came close to 4-logic. For failures, all nearly-passing logical trees are thus dumped; if two or more logics are "close" at one shift position, the format is analogous to that of Fig. 7(d).

In Fig. 7(e), the tape images of the preceding four characters are displayed with horizontal and vertical guide indexes. (Every other alphabetic index has been omitted for legibility.) These images are exactly the data which the simulator has just processed and described. For example, one notes that the successful "3" of Fig. 7(a)—image 1— is a reasonably recognizable character, visually. The next image—the "4" and "□" conflict—certainly is aberrated from the normal conception of a 4. The remaining two "4"s are visually identifiable; since they came closer to passing the logic for "□" than for "4," both 4-logic and □-logic should be altered if this sample is to be properly recognized.

The value of individual-test summary statistics becomes increasingly clear as one attempts to alter recognition logic to fit the requirements of a group of samples. The individual-test dumps for each sample can show only the reaction of the logic to this one sample—valuable in the earliest stages of logic design, but difficult to digest for an aggregate of samples.

Acknowledgment

The author wishes to express his gratitude to Dr. Y. M. Hill of the IBM General Products Division Development Laboratory, for his many invaluable suggestions both for the mechanism of SCRIPT and for sophisticated information retrieval techniques in the output of the program.

An Incremental Computer Technique for Solving Coordinate-Rotation Equations*

C. S. DEERING†, MEMBER, IRE, AND C. B. SHELMAN†

Summary—A method for solving coordinate rotation equations on an incremental digital computer is described in this paper. The method employs the basic incremental techniques used in a digital differential analyzer (DDA). However, the method differs from the usual DDA approach in that it takes full advantage of the possibilities for combining "remainder," or "*R*" registers. This results in a reduction of digital storage-capacity requirements. Further, techniques are employed which, because of the specific nature of the application, permit additional reductions in storage requirements.

THE incremental digital computer is well adapted to the solution of simultaneous differential equations.[1–3] High digital accuracy is obtained even though relatively simple arithmetic and control circuitry are required. These characteristics may be employed to advantage in the solution of coordinate-system rotation equations.

Fig. 1 shows two orthogonal coordinate systems,

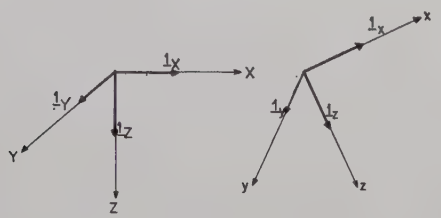


Fig. 1—*XYZ* and *xyz* axis systems.

XYZ and *xyz*. The orientation of either axis system with respect to the other is specified by the following direction cosine matrix:

$$\begin{pmatrix} \underline{1}_X \\ \underline{1}_Y \\ \underline{1}_Z \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \begin{pmatrix} \underline{1}_x \\ \underline{1}_y \\ \underline{1}_z \end{pmatrix}, \tag{1}$$

where the R_{ij} are direction cosines, and $\underline{1}_X$, $\underline{1}_Y$, $\underline{1}_Z$, $\underline{1}_x$, $\underline{1}_y$, and $\underline{1}_z$ are unit vectors along the axes indicated by the subscripts. Both the *XYZ* and the *xyz* systems may rotate independently with respect to inertial space. A classical method[4] may be employed to differentiate (1) with respect to time and derive the following set of differential equations:

$$\dot{R}_{11} = R_{12}\omega_z - R_{13}\omega_y + R_{21}\omega_Z - R_{31}\omega_Y \tag{2}$$
$$\dot{R}_{12} = R_{13}\omega_x - R_{11}\omega_z + R_{22}\omega_Z - R_{32}\omega_Y \tag{3}$$
$$\dot{R}_{13} = R_{11}\omega_y - R_{12}\omega_x + R_{23}\omega_Z - R_{33}\omega_Y \tag{4}$$
$$R_{21} = R_{22}\omega_z - R_{23}\omega_y + R_{31}\omega_X - R_{11}\omega_Z \tag{5}$$
$$R_{22} = R_{23}\omega_x - R_{21}\omega_z + R_{32}\omega_X - R_{12}\omega_Z \tag{6}$$
$$R_{23} = R_{21}\omega_y - R_{22}\omega_x + R_{33}\omega_X - R_{13}\omega_Z \tag{7}$$
$$R_{31} = R_{32}\omega_z - R_{33}\omega_y + R_{11}\omega_Y - R_{21}\omega_X \tag{8}$$
$$R_{32} = R_{33}\omega_x - R_{31}\omega_z + R_{12}\omega_Y - R_{22}\omega_X \tag{9}$$
$$R_{33} = R_{31}\omega_y - R_{32}\omega_x + R_{13}\omega_Y - R_{23}\omega_X. \tag{10}$$

In the above equations the quantities ω_x, ω_y, and ω_z are the angular velocity components of the *xyz* axis system. The quantities ω_X, ω_Y, and ω_Z are the angular velocity components of the *XYZ* axis system. The "*R-matrix*" $[R_{ij}]$ of (1) may be determined by solving (2)–(10), in which the six $\omega_k (k = x, y, z, X, Y, Z)$ are treated as the independent variables.

Incremental Equations

Incremental techniques are employed in the solution of (2)–(10). It is first necessary to write the equations in differential form. In (2)–(10) $d\mu_k/dt$ is substituted for each ω_k (where k denotes the axis), and each equation is multiplied through by dt (an increment of time). The following set of equations results:

$$dR_{11} = R_{12}d\mu_z - R_{13}d\mu_y + R_{21}d\mu_Z - R_{31}d\mu_Y \tag{11}$$
$$dR_{12} = R_{13}d\mu_x - R_{11}d\mu_z + R_{22}d\mu_Z - R_{32}d\mu_Y \tag{12}$$
$$dR_{13} = R_{11}d\mu_y - R_{12}d\mu_x + R_{23}d\mu_Z - R_{33}d\mu_Y \tag{13}$$
$$dR_{21} = R_{22}d\mu_z - R_{23}d\mu_y + R_{31}d\mu_X - R_{11}d\mu_Z \tag{14}$$
$$dR_{22} = R_{23}d\mu_x - R_{21}d\mu_z + R_{32}d\mu_X - R_{12}d\mu_Z \tag{15}$$
$$dR_{23} = R_{21}d\mu_y - R_{22}d\mu_x + R_{33}d\mu_X - R_{13}d\mu_Z \tag{16}$$
$$dR_{31} = R_{32}d\mu_z - R_{33}d\mu_y + R_{11}d\mu_Y - R_{21}d\mu_X \tag{17}$$
$$dR_{32} = R_{33}d\mu_x - R_{31}d\mu_z + R_{12}d\mu_Y - R_{22}d\mu_X \tag{18}$$
$$dR_{33} = R_{31}d\mu_y - R_{32}d\mu_x + R_{13}d\mu_Y - R_{23}d\mu_X. \tag{19}$$

* Received by the PGEC, April 3, 1961.
† Vought Electronics, a division of Chance Vought Corporation, Arlington, Tex.
[1] G. F. Forbes, "Digital Differential Analyzers, an Application Manual for Digital and Bush Type Differential Analyzers," Pacoima, Calif.; 1956.
[2] M. Palevsky, "The design of the Bendix digital differential analyzer," PROC. IRE, vol. 41, pp. 1352–1356; October, 1953.
[3] R. K. Richards, "Arithmetic Operations in Digital Computers," D. Van Nostrand Co., Inc., New York, N. Y.; 1955.
[4] See, for example, H. C. Corben and P. Stehle, "Classical Mechanics," John Wiley and Sons, Inc., New York, N. Y.; 1950.

In the above equations $d\mu_k$ may be interpreted as an incremental rotation of the respective axis system about the k axis; dR_{ij} is the incremental change in R_{ij}.

Computer Mechanization of the Equations

In solving the incremental equations, the following relationship is implied:

$$(R_{ij})_n = (R_{ij})_{n-1} + (dR_{ij})_n, \qquad (20)$$

where $(R_{ij})_{n-1}$ and $(R_{ij})_n$ are successive values of R_{ij} before and after the incremental change dR_{ij} occurs. An iteration technique is used to compute the R_{ij}. During each cycle of the iteration process the nine dR_{ij} are each computed from (11)–(19), and the nine R_{ij} are incremented as in (20). During any iteration cycle, the dR_{ij} are computed in sequence. The arithmetic operations specified by (11)–(20) are performed serially by a synchronous computer.

Control Functions

Control functions are necessary to effect the solution of the incremental equations. The control functions are simply a group of signals which identify the sequence of clock pulses within the computing cycle. The group of logical and storage elements which are programmed to perform these functions is referred to as the *control unit*. The control unit produces the following signals: D_1, D_2; C_1, C_2, C_3; W_1, W_2, W_3; B_1, B_2, $\cdots$, B_{22}. Each computing cycle is divided into two equal periods D_1 and D_2. Both D_1 and D_2 are equally divided by the three periods denoted C_1, C_2, and C_3. Periods C_1, C_2, and C_3 are each equally divided by the "words" W_1, W_2, and W_3. Finally, the words consist of twenty-two "bits" B_1, B_2, $\cdots$, B_{22}. A logical "1" occurs successively on B_1, B_2, $\cdots$, B_{22} at the computer clock frequency which is 1 Mc. This information is diagrammed in Fig. 2. Each of the control functions is available at a definite location in the computer. The control-function sequence is repeated once each computing cycle.

Equation Solution

The incremental equations are solved by the method indicated in the block diagram of Fig. 3. Some of the aspects of this method have been discussed in the literature.[5] Two magnetostrictive delay lines are shown near the bottom of Fig. 3. Delay line 1 stores the nine-direction cosines of the R-matrix. Each R_{ij} is nineteen bits in length. Since the word length is twenty-two bits, three of the twenty-two are blank. Two's complement number representation is employed. The sign is determined by the value of the most significant bit. The positions of the R_{ij} are shown as they would exist at

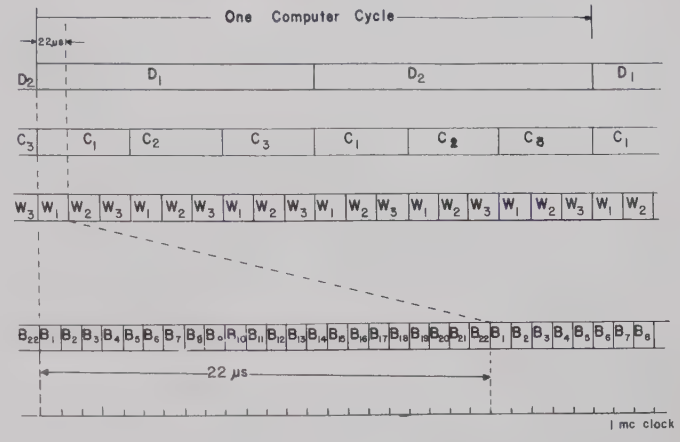


Fig. 2—Control functions.

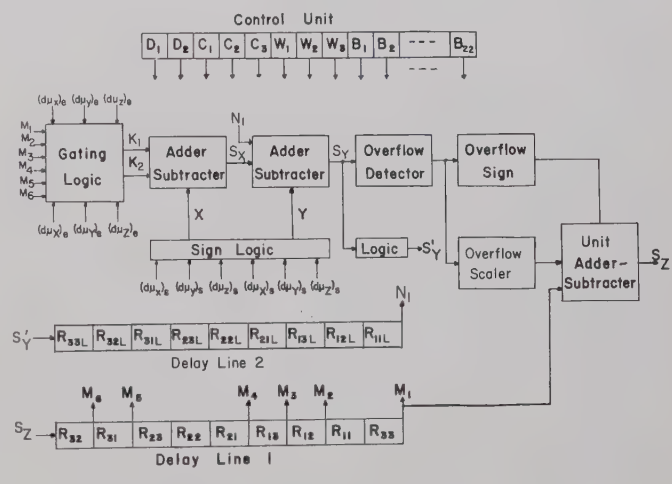


Fig. 3—Incremental computer block diagram.

the beginning or at the center of a computing cycle. At each clock pulse the R_{ij} advance one bit position toward the right. The information is inserted into the write amplifier at the left end of the line as S_Z, "picked off" at read amplifier M_1, operated on by the computer logic and reinserted as S_Z, etc. Thus, the information circulates continuously with the timing such that exactly one-half a computing cycle is required for a particular R_{ij} to complete its journey around the delay-line loop. Intermediate read amplifier pick-offs M_2, M_3, M_4, M_5 and M_6 are located appropriately as required to solve (11)–(19).

Delay line 2 operates in a manner similar to that of delay line 1. It circulates the "remainder" portion R_{ijL} of each R_{ij}. The R_{ijL} are eighteen bits in length. Notice that the R_{ijL} are advanced one word time from the respective R_{ij}. No intermediate pick-offs are required on delay line 2.

During each word time half of one of the equations of the set (11)–(19) is solved. For example, during $D_1C_1W_1$ the first two terms on the right side of (11) are used to form dR_{11} which is added to R_{11L}. At $D_2C_1W_1$,

[5] R. E. Sprague, "Fundamental concepts of digital differential analyzer method of computation," *Mathematical Tables and Other Aids to Computation*, vol. 6, pp. 41–49; January, 1952.

which occurs exactly nine word-times later, the third and fourth terms on the right side of (11) form dR_{11} which is likewise added to R_{11L}. At $D_1C_1W_2$ the first two terms of (12) are computed to form dR_{12} which is added to R_{12L}. The same procedure is used for all nine equations. D_1 and D_2 last for exactly nine words each. Since it requires nine word times for the R_{ijL} to circulate through the delay line, a particular R_{ijL} is available at the end of the line twice during each computing cycle, once during D_1 and once during D_2. During the nine word-times of D_1, the first two terms of (11)–(19) are solved; during the nine word-times of D_2, the second two terms are solved so that during each computing cycle all the terms of (11)–(19) are solved.

The independent variables employed in the solution are the six $d\mu_k$ increments. Each of these increments, as scaled for the computer, may take on one of three values: $+1$, -1, or zero. These three values may be represented by two binary digits, one denoting the existence of $d\mu_k$, $(d\mu_k)_e$ and the other denoting the sign $(d\mu_k)_s$.

The first step in solving half of one of the equations is to gate the two R_{ij} of the equation being solved into an adder-subtracter. The R_{ij} are gated by the two $(d\mu_k)_e$ specified in the equation. These R_{ij} are available at the various pick-off points on delay line 1. The gated R_{ij} are denoted K_1 and K_2 in Fig. 3. During D_1, K_1 is derived from the first right-hand term in each of (11)–(19) and K_2 from the second term. K_2 is added to or subtracted from K_1 to form S_X. The binary variable X determines whether addition or subtraction is performed.

S_X is added to or subtracted from N_1 in a second adder-subtracter to form S_Y. The choice of addition or subtraction is controlled by Y. During any particular word time, N_1 is the remainder portion of the R_{ij} being up-dated during that word time; *i.e.*, dR_{ij} is added to R_{ijL}. X and Y are derived from the two $(d\mu_k)_s$ associated with K_1 and K_2.

Most of the remaining operations indicated in Fig. 3 are concerned with handling the overflows from the R_{ijL} which may occur when dR_{ij} is added to R_{ijL}. Round-off of the R_{ij} is implemented by inserting a "1" prior to the start of computation into the most significant bit (which is the 2^{-1} bit) of each R_{ijL}. This inserted bit biases an R_{ijL} to the midpoint of its range, so that the associated R_{ij} are always rounded to the least significant bit.[6] Since two's complement number representation is used, any overflow from an R_{ijL} is detected by the presence of a "1" in the 2^0 bit position of S_Y. When a "1" occurs in the 2^0 position it is deleted from S_Y to form S_Y', which is inserted back into delay line 2. S_Y' is the new R_{ijL} after dR_{ij} has been added. The sign of an overflow is determined logically. When a positive overflow occurs, a "1" is added to the least significant bit of R_{ij}; a "1" is subtracted for a negative overflow. The unit adder-subtracter is used for this purpose. The position of the least significant bit is determined by the overflow scaler in Fig. 3. After the overflow bit has been added to or subtracted from R_{ij}, the "updated" R_{ij} is applied as S_Z to the write amplifier on the left end of delay line 1. Thus, the R_{ij} are recirculated in delay line 1. Initial conditions are established for the R_{ij} prior to computation.

The computer is designed to process information serially, *i.e.*, one bit at a time. The least significant bit is processed first and the most significant, last. Consequently, in the delay lines the least significant bit of a word is stored, or read, first followed by bits of increasing significance.

Computer Solutions

As the xyz and XYZ axis systems rotate, the angular increments of rotation, $d\mu_x$, $d\mu_y$, $d\mu_z$, $d\mu_X$, $d\mu_Y$, and $d\mu_Z$ are applied to the computer. Each increment may occur at a maximum rate of once per computer cycle. A computer cycle lasts 396 μsec. The size of the increment is a function of the R_{ij} scaling and of the overflow scaling. In the computer, two's complement number representation is used, and the R_{ij} are restricted so that

$$1 > R_{ij} \geq -1. \tag{21}$$

In order to insure compliance with (21), the R_{ij} are scaled by one-half; for example, when a direction cosine has the value unity, the R_{ij} in the computer is one-half. In the application in which the computer is used, the $d\mu_k$ increments are 2^{-18} radian (positive or negative, depending upon the direction of rotation).

As an example of a computer solution of a coordinate rotation, refer to Fig. 4. In Fig. 4 the xyz and XYZ axis systems are aligned. The R matrix has "ones" along the major diagonal. Corresponding to this condition, the initial conditions

$$R_{11} = R_{22} = R_{33} = 1/2$$

and

$$R_{12} = R_{13} = R_{21} = R_{23} = R_{31} = R_{32} = 0$$

are inserted in delay line 1. "Ones" are inserted into the most significant bit portions of the R_{ijL} for round-off purposes. Computation is started, and on each computing cycle, $d\mu_x$, $d\mu_y$, and $d\mu_z$ increments are applied until the xyz system rotates to the position shown in Fig. 5 with x, y, and z aligned with Y, Z, and X, respectively. A total of 316,984 increments on each of $d\mu_x$, $d\mu_y$, and $d\mu_z$ are required. After rotation, the R_{ij} in the computer are compared with the values shown in the matrix in Fig. 5. The maximum error in any R_{ij} is approximately one part in 250,000. The rotation requires approximately 126 seconds.

[6] M. L. Klein, F. K. Williams and H. C. Morgan, "Digital differential analyzers," *Instruments and Automation*, vol. 30, pp. 1105–1109; June, 1957.

$$[R_{ij}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

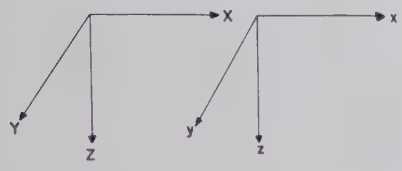


Fig. 4—Axis systems aligned.

$$[R_{ij}] = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

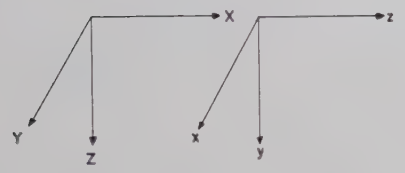


Fig. 5—Axis systems after rotation of *xyz* system.

Comparison of Storage Requirements with Those of a General-Purpose DDA

A general-purpose DDA (digital differential analyzer) may be used to solve (11)–(19). In a DDA the basic computing unit is the integrator which solves for Z in the equation

$$Z = \int Y dX, \tag{22}$$

by summing finite differentials as indicated in the equation

$$Z \approx \sum Y dX. \tag{23}$$

Each integrator contains a register, called the Y register, which stores the integrand Y. When an incremental change dX occurs in X, the number in the Y register is added to that in a second register which is denoted the R register. The R register stores the least significant, or "remainder," portion of $\sum YdX$. The bit capacity of the R register is limited, and the "overflow" of the R register may be accumulated in the Y register of another integrator. This accumulated overflow is the most significant portion of $\sum YdX$. Thus, each DDA integrator contains two registers, a Y register and an R register, and a means for adding the Y register to the R register when dX occurs.

One integrator of a general-purpose DDA is required for each of the terms on the right side of (11)–(19). For example, from (11),

$$R_{11} \approx \sum dR_{11} = \sum R_{12} d\mu_z - \sum R_{13} d\mu_y + \sum R_{21} d\mu_Z - \sum R_{31} d\mu_Y. \tag{24}$$

Thus, four integrators are required to compute R_{11}. It follows that thirty-six integrators are needed in the solution of all nine R_{ij}.

The overflows from the R registers of the four integrators used in solving (11) are accumulated to form the most significant portion of R_{11}. It is necessary to temporarily store these overflows until the arithmetic unit is ready to add them to the register containing R_{11}. The same holds true for the other eight R_{ij}. Thus, temporary storage is required for the overflows in addition to the seventy-two storage registers of the thirty-six integrators.

In the mechanization of the equations by the method of Fig. 3, the storage requirements are reduced. Instead of using separate Y and R registers for each right-hand term in (11)–(19), only nine Y and nine R registers are used. The nine Y registers are contained in delay line 1 of Fig. 3. The nine R registers are contained in delay line 2. Each R register may have the contents of four Y registers added to it, so that in a sense, four R registers are combined. The storage requirements for Y and R registers are reduced from seventy-two words to eighteen words, a factor of four to one.

The R_{ij} in delay line 1 are precessed one word time from the R_{ijL} in delay line 2, so that, whenever an overflow occurs from the most significant bit position of an R_{ijL}, the R_{ij} associated with it may immediately accept this overflow bit. This removes the necessity for storing a large number of overflow bits because each overflow is used immediately after it is generated.

Conclusion

An incremental computer has been described which solves a set of coordinate rotation equations. Special techniques are employed so that the storage requirements have been reduced by a factor of four from those of a general-purpose DDA. These techniques can be employed in other special-purpose incremental computers to effect comparable reductions in storage requirements.

Acknowledgment

The authors wish to acknowledge the contributions of J. R. Campbell and G. Cocharo in suggesting the incremental approach and deriving the incremental coordinate-rotation equations.

Two-Level Correlation on an Analog Computer*

C. L. BECKER†, MEMBER, IRE, AND J. V. WAIT‡, MEMBER, IRE

Summary—It has been known for some time that an approximate correlation analysis of a random process can be performed using quantized values of the signal. The simplest form possible is a two-level correlation, wherein merely the polarities of the process at two sampling times, T_1 and $T_2=T_1+\tau$ are compared. This report describes a study of this technique, using analog-computer circuitry; it is an interesting example of how existing electronic analog computers can implement essentially digital functions in making accurate statistical measurements.

I. Introduction

ONE extremely important measurement associated with the application of statistical techniques to analog simulation studies is the determination of correlation functions. In many cases, the specialized equipment required for performing correlation measurements is not available at a small analog-computer facility.

An approximate correlation analysis of a stationary random process can be performed using quantized values of the signal [1], [2]. The simplest form possible is a two-level correlation. In this method, the original random process is passed through a clipper or comparator as the first step in the correlation analysis. The resulting two-level function retains the same zero crossings as the original process but, of course, contains no information about the amplitude of the original process, except its polarity. Nevertheless, a correlation measurement made upon the two-level process yields much useful information about the original process, and it can be performed in a relatively simple manner.

II. System Description

Fig. 1 shows the elements of a one-bit autocorrelator, constructed primarily with analog-computer circuitry; circuit details are shown in Fig. 7. The principal elements of the system are:

1) A source of 100-volt push-pull pulses (designated the sampling-pulse generator in Fig. 1). These can be easily derived from a square-wave generator; they provide the signals necessary for resetting the timing circuit and operating the holding comparator. The square-wave period determines the sampling rate.
2) An optional input comparator, which is used to provide a preamplified and limited input signal at point C. This circuit is not essential to the operation of the correlator but provides improved high-speed operation when using relatively slow operational amplifiers.
3) An analog multivibrator or holding comparator, which is used to store the polarity of the input random process $X(t)$ at the time T_1 (this value, which will be designated $[X_1]$, appears at point E).
4) A second free-running comparator, whose output (point D) is also either plus or minus 100 volts depending upon the instantaneous polarity of $X(t)$. Its output is thus a two-level random signal designated $[X(t)]$.
5) A timing circuit, which is reset at time T_1, and which produces a rapid positive rise at time $T_2=T_1+\tau$. This circuit requires one operational amplifier. The output (point G) is differentiated to produce a positive pulse for injection into the output circuit.
6) A logic circuit, requiring six diodes and a triode, whose output F is the binary product of the two comparator outputs, D and E; that is $F=DE+\overline{D}\,\overline{E}$.
7) An output gating circuit, a simple voltage divider, whose output H is positive only if $[X_1]$ and $[X(t)]$ are correlated at the time when the T_2 timing pulse is generated.
8) Two counters, one for counting the total sample or run length, and the other for making the correlation count. The correlation counter registers one count each time the correlator output signal H momentarily becomes positive. The run-length counter registers one count for each sampling interval.

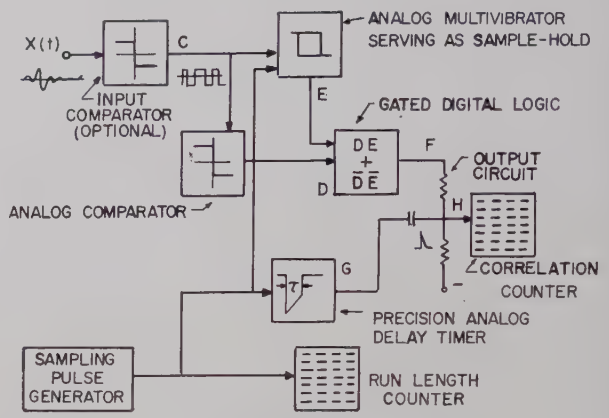


Fig. 1—System block diagram.

When built up on a typical analog computer, the system requires six operational amplifiers, two triodes, and

* Received by the PGEC, April 30, 1960.
† Dept. of Elec. Engrg. University of Santa Clara, Santa Clara, Calif. Formerly Dept. of Elec. Engrg., University of Arizona, Tucson, Ariz.
‡ Dept. of Elec. Engrg., University of Arizona, Tucson, Ariz.

fifteen diodes. A thyratron driving circuit was also constructed for each counter. This circuit should initiate one count each time the input signal momentarily becomes positive. Out of the six operational amplifiers used, one is the optional input comparator, and two more are used to obtain the 100-volt push-pull timing signals. Only three operational amplifiers are essential elements of the system.[1]

III. Basic Sequence of Operation

Fig. 2 shows typical waveforms associated with the correlator operation. Time is measured from a positive-going transition of the square-wave timing voltage A (see Fig. 7). In the case shown, the sampling period T is one second (corresponding to a 1 cps setting on the square-wave generator); the example correlation time τ is 0.2 second. The sequence of operations is as follows:

1) When the square-wave timing voltage A swings positive, the feedback loop in the analog multivibrator is turned on, and the output E is held. The timing generator (waveform G) is reset to a negative level at the same time. (Square-wave B drives the opposite side of the diode gate in the multivibrator feedback loop.) In the example shown, the above events occur at $t=T_1=nT$; $n=0, 1, 2, \cdots$; *i.e.*, the beginning of a sampling interval.
2) The zero crossings of the input are sensed by the input comparator, whose output is the two-level process shown at C.
3) Waveform D, the output of the free-running comparator has the same polarity and zero crossings as the input random process $X(t)$.
4) The multivibrator is held for one half a sampling interval. It is brought out of hold when the square-wave A swings negative.
5) The outputs of the two comparators (waveforms D and E) are combined in the logic circuit to form the binary product, $F=DE+\overline{D}\overline{E}$. This signal is positive if, and only if, D and E are both of the same polarity.
6) At time $T_2=nT+\tau$, the output of the timing circuit (waveform G) swings positive. This voltage is differentiated, and the resulting gate pulse is added, along with a negative bias, to the output of the logic circuit. This sum appears at the output of the correlator (waveform H). Note that H is positive only if F is $+100$ when time T_2 occurs. Each time H goes positive, one count is registered in the correlation counter.

In the example shown, during the first sampling interval the value of $X(t)$ at $t=\tau=0.2$ has the same polarity $(+)$ as it does at $t=0$. Therefore, the output of the logic circuit (F) is positive at $t=0.2$, and one count is registered. This corresponds to a positive correlation for the first sample.

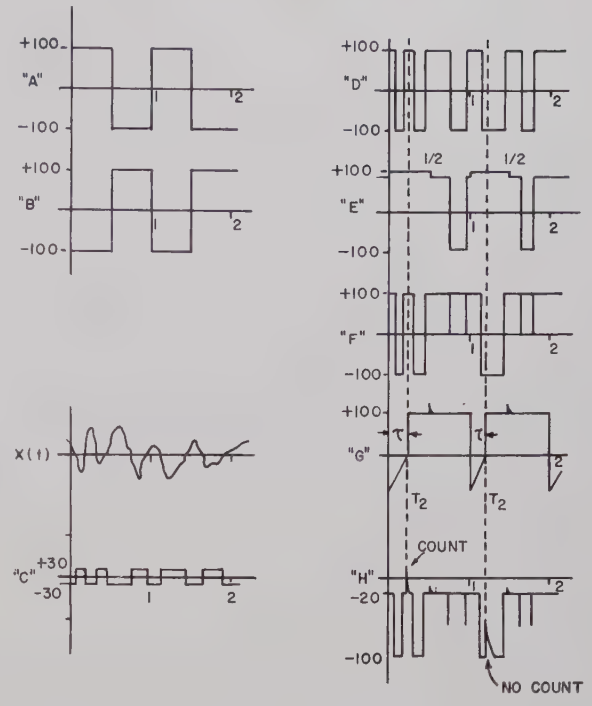


Fig. 2—Correlator waveforms.

At the beginning of the second sampling interval $(t=1)$, the polarity of the random process is again positive, but it is negative τ seconds later $(t=1.2)$. Thus the output at H is not driven positive by the timing pulse, and no correlation count is registered.

During the two sampling intervals diagrammed, the run-length counter has registered a count of 2, while the correlation counter has registered a count of 1. The sequence is continued until the desired number of samples have been taken for the given value of τ.

IV. Analysis of Two-Level Autocorrelation Results

To demonstrate the results which can be obtained with a two-level autocorrelator, the following signals were used as inputs to the correlator:

1) a 10 cps sine wave,
2) random noise with a 10-radian/sec bandwidth,
3) random noise with a 20-radian/sec bandwidth,
4) a 10-cps sine wave plus random noise with a 10-radian/sec bandwidth.

Data were taken for values of τ up to 0.2 second. A total run length of 2000 samples was taken at each experimental point. No detailed analysis of the variance in sample counts was made but, in general, it appeared that for the input functions analyzed, a run of 1000 or 2000 samples provides a sufficiently converged mean count rate. The number of samples in which $[X_1]$ and $[X_2]$ were correlated was counted by the correlation

[1] If the input random process has zero average value, a further simplification would be possible which would eliminate one triode and four of the diodes (see Section V).

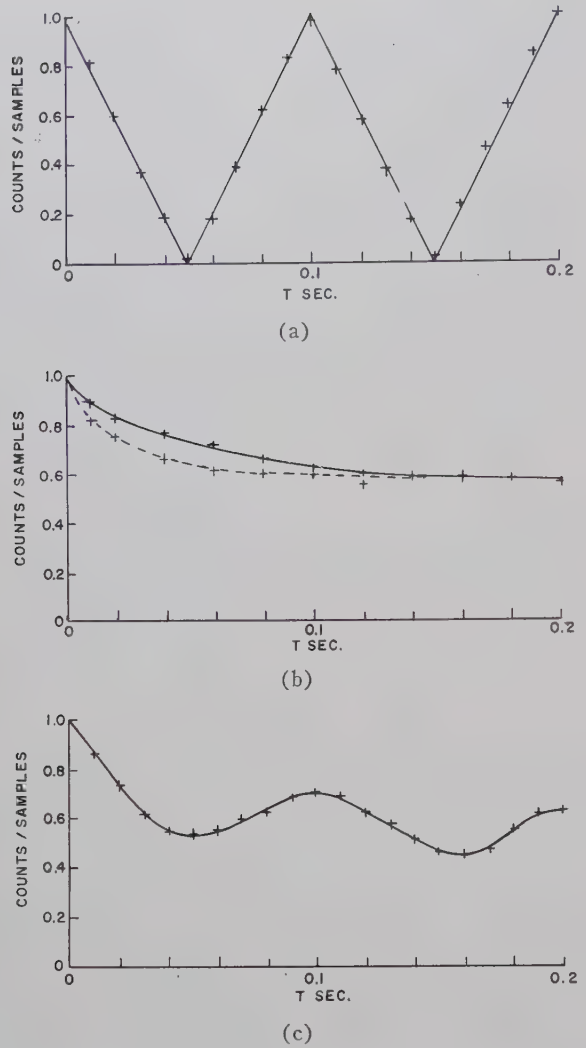


Fig. 3—Two-level autocorrelator output: (a) 10-cps sine wave, (b) random noise with a 10-radian/sec bandwidth (solid line) and with a 20-radian/sec bandwidth (dashed line), (c) 10-cps sine wave plus noise in 10-radian/sec bandwidth.

counter. For each input process, the ratio of these counts to the total number of samples is plotted in Fig. 3. When the input process has zero mean value, the corresponding autocorrelation coefficients are found by

$$\text{Autocorrelation coefficient} = 2 \times \left(\frac{\text{correlated counts}}{\text{total samples}} - 1/2 \right)$$

and are shown in Figs. 4 and 5.

In each case, the exact autocorrelation function has been computed, as well as the ideal two-level autocorrelation function. These results have been plotted on the same graph in each case in order to give a direct pictorial presentation of how much the simple two-level autocorrelation function can be expected to depart from the exact function. In addition, the experimental data has been plotted on the same graphs to demonstrate how closely the machine performance approaches the ideal two-level function.

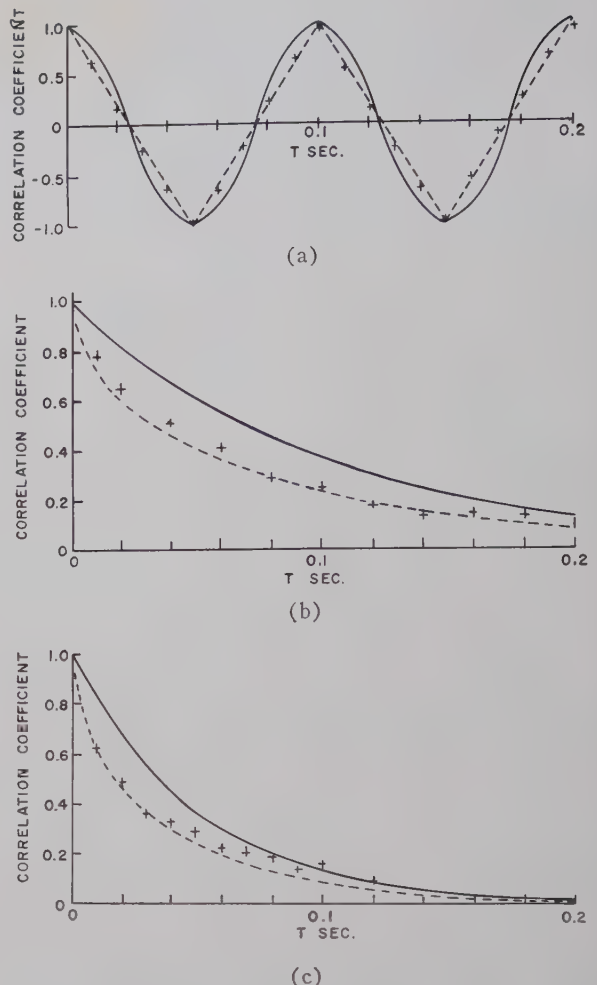


Fig. 4—Autocorrelation functions: (a) 10-cps sine wave, (b) random noise with a 10-radian/sec bandwidth, (c) random noise with a 20-radian/sec bandwidth. In each case, the solid line represents the exact autocorrelation function, the dashed line represents the theoretical two-level autocorrelation function, and + represents an experimental point.

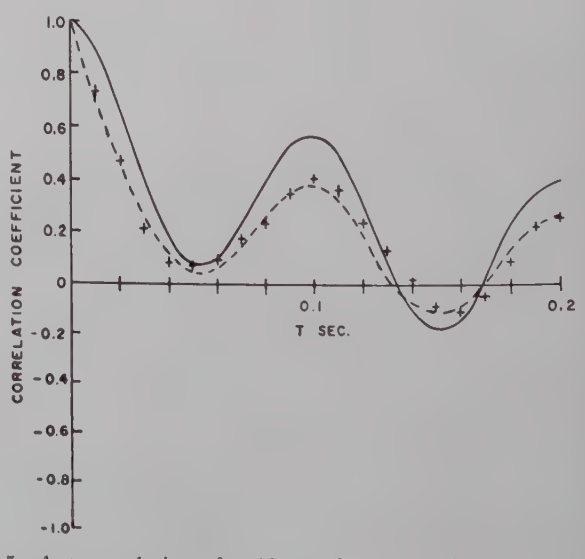


Fig. 5—Autocorrelation of a 10-cps sine wave plus noise in a 10-radian/sec bandwidth. The solid line represents the exact autocorrelation function, the dashed line represents the theoretical two-level autocorrelation function and + represents an experimental point.

Sine-Wave Input

In the case of the 10-cps sine wave, the exact autocorrelation function can be found from the well-known relationship

$$R(\tau) = \frac{1}{T}\int_{-T/2}^{T/2} f(t)f(t+\tau)dt.$$

After substituting and integrating, we have

$$R(\tau) = \cos 62.8\tau.$$

Therefore, the autocorrelation function of a sine wave is a cosine wave of the same frequency, which is a well-known result.

In an ideal two-level autocorrelator, a sine wave is reduced to a square wave of period equal to that of the sine wave, and then correlated. In a manner similar to the above, it can be shown that the two-level autocorrelation of any periodic function is a triangular wave of frequency equal to that of the original function, since all periodic functions are reduced to periodic square waves in an ideal two-level correlator.

In Fig. 4(a), it can be seen that the experimental results agree very closely with the ideal triangular waveform which was derived for the two-level case.

Random Noise Input

The random noise source consisted of a random telegraph wave passed through a first-order low-pass filter as shown in Fig. 6. In the Appendix, it is shown that the exact autocorrelation function of the filtered random telegraph wave is given by

$$R_{yy}(\tau) = e^{-\omega_c|\tau|}.$$

It is also shown that the two-level autocorrelation function for the same input is given by

$$R_{yy}(\tau) = (2/\pi)\sin^{-1}[e^{-\omega_c|\tau|}].$$

These results are plotted in Fig. 4(b) and 4(c) for $\omega_c = 10$ and 20 radians/sec, respectively. It can be seen that the experimental results agree quite well with these latter curves. In both cases, there is a tendency for the experimental results to approach a non-zero value of correlation coefficient for larger values of delay-time τ. This is probably caused by the presence of a small dc voltage in the output of the noise generator, and to compare offsets.

Sine Wave Plus Random Noise Input

For the case of a sine wave plus random noise, a random telegraph wave and a sine wave were combined in the input to a low-pass filter as shown in Fig. 6. In the Appendix it is shown that the exact autocorrelation function is given by

$$R_{yy}(\tau) = \frac{e^{-\omega_c|\tau|} + a^2\cos\omega_s\tau}{1+a^2}.$$

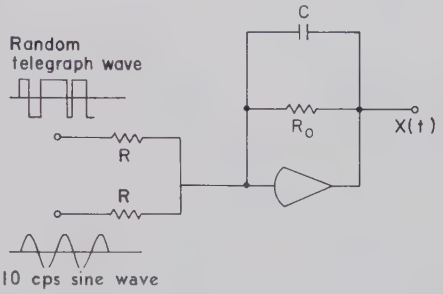


Fig. 6—Input mixing and filtering amplifier.

The relative amplitudes of the sine wave and the noise at the filter output were adjusted so that $a^2 \approx \frac{1}{2}$. This ratio was only approximate, since the rms amplitude of the noise was not measured precisely. With $a^2 = \frac{1}{2}$, we have

$$R_{yy}(\tau) = \frac{e^{-\omega_c|\tau|} + 1/2\cos\omega_s\tau}{3/2}.$$

This function is plotted in Fig. 5 for $\omega_c = 10$ radians/sec and $\omega = 62.8$ radians/sec.

Using only the first term of McFadden's expression[2] for the two-level autocorrelation function, we have

$$r(\tau) = 2/\pi\sin^{-1}\frac{e^{-\omega_c|\tau|} + 1/2\cos\omega_s\tau}{3/2}.$$

This function is also plotted in Fig. 5 for the same values of ω_s and ω_c. It can be seen that the experimental points agree quite well with this latter function. The tendency toward larger discrepancies at larger values of delay time (τ) may be due in part to neglecting the second term in McFadden's expression for the two-level autocorrelation function.

It is interesting to note that the shape of the two-level autocorrelation function is essentially similar to the shape of the exact autocorrelation function in the cases where noise was present. In Fig. 5, the principal difference is that the exact autocorrelation function has a different slope near $\tau = 0$, and its variations are greater in amplitude.

V. Circuit Details of the One-Bit Correlator

Fig. 7 shows the analog-computer circuitry required for the one-bit correlator. All operational amplifiers are chopper stabilized (Philbrick USA-3's were used in the experimental setup). Both triodes are type 12AU7, and the diodes were type 1N643. The operation of the three comparators (amplifiers *C*, *D*, and *E*) and the timing circuit (amplifier *G*) have been described in previous articles [3].

Holding Comparator or Multivibrator

The feedback loop around the holding comparator makes this circuit operate as a bistable multivibrator

[2] See the Appendix.

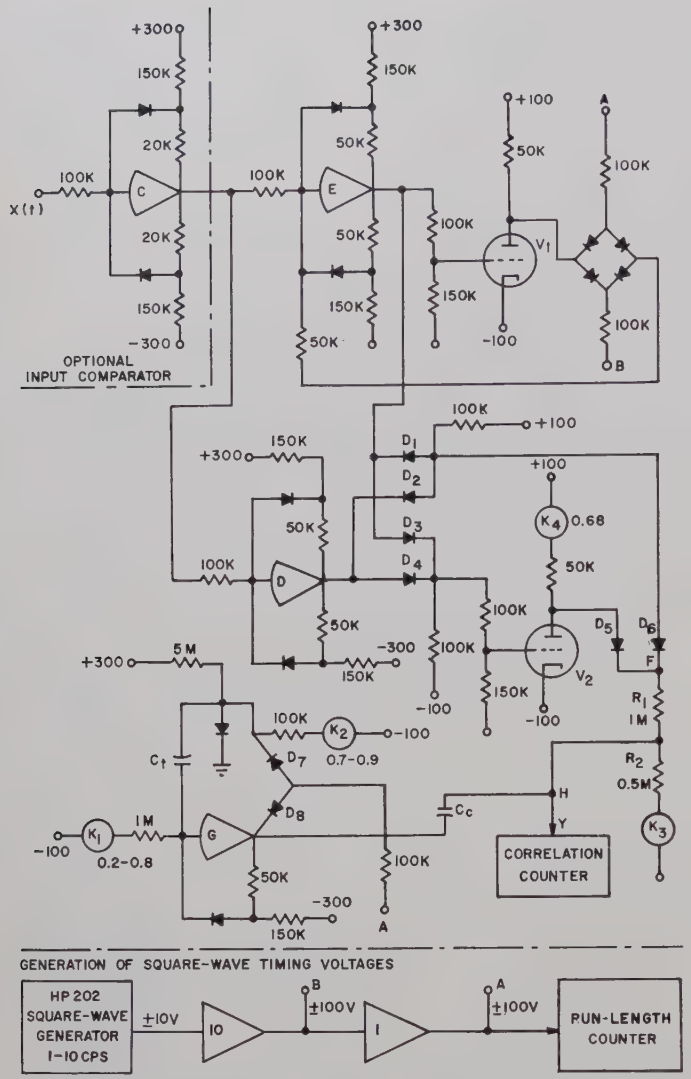


Fig. 7—Correlator circuit.

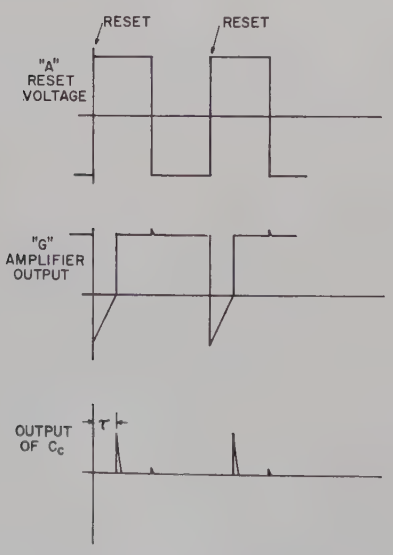


Fig. 8—Timing circuit waveforms.

when the diode gate is turned on (at time T_1), thus storing the comparator state then present.

Logic Circuit

The output of comparators D and E are compared in the logic circuit to form: $F=DE+\overline{D}\overline{E}$.[3] This operation is formed as follows:

1) diodes D_1 and D_2 form DE (AND gate),
2) diodes D_3 and D_4 form $D+E$ (OR gate),
3) the output of the inverter triode (V_2) is then $\overline{D+E}=\overline{D}\overline{E}$,
4) diodes D_5 and D_6 form $DE+\overline{D}\overline{E}$ (OR gate).

Timing Circuit (See Figs. 7 and 8)

The timing circuit (amplifier G) is reset by square-wave-voltage A. The timing interval begins when voltage A swings positive, bringing the diode $\frac{1}{2}$-gate (D_7 and D_8) out of the OFF condition. The initial reset voltage V_0, is determined by the setting of potentiometer K_2, and is normally about -80 volts. The slope of the ramp portion is determined by the setting of potentiometer K_1. With $C_t=0.01$ mf, the time interval $0.01<\tau<0.2$ sec can be covered by adjusting K_1 alone. The output of the timing generator is differentiated by C_c. The value of C_c must be changed as τ is varied. For $0.01<\tau<0.05$, 0.001 mf was used, and for $\tau>0.05$, 0.003 mf was used.

Output Gating Circuit

The outputs of the logic circuit and the timing generator are mixed by network R_1 and R_2. Potentiometer K_3 is adjusted so that the differentiated timing pulses drive point H approximately 2 or 3 volts positive when point F is at $+100$ volts (*i.e.*, D and E of the same sign). K_3 is normally set at about 0.8.

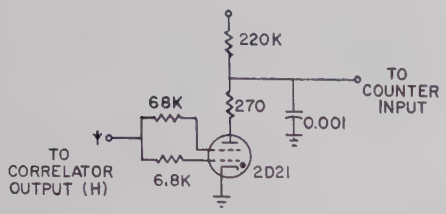
Counter Driving Circuits (See Fig. 9)

Two four-decade Berkeley counters were modified to include thyratron trigger circuits, to provide fast trigger pulses. The basic circuit shown in Fig. 9(a) was used in the correlation counter. The thyratron conducts whenever the input (point Y) and consequently the grid become positive. A fast negative-going pulse is generated at the plate and coupled to the first counter stage through an internal capacitor in the counter unit. The 0.001 mf capacitor assists in turning the thyratron off, and appears to be an optimum value to prevent double triggering. The thyratron is returned to the nonconducting state when the correlator output signal becomes negative, following the registering of a count. Fig. 9(b)

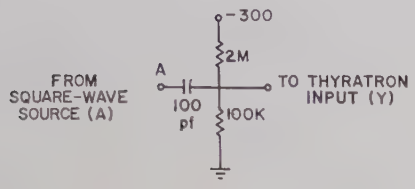
[3] If the input random process has zero average value, we need only $F=DE$, which could be formed using only two diodes in an "AND" gate. Using this method, then we have

$$\text{Autocorrelation coefficient} = 2\times\left(\frac{2\times(+\text{correlation counts})}{\text{total samples}}-\frac{1}{2}\right).$$

This simplification might lead to additional errors in the resulting correlation measurements due to offsets in the comparators, etc.



(a)



(b)

Fig. 9—Counter driving circuit: (a) basic circuit, (b) bias circuit.

shows a biasing and coupling network which was added to the thyratron circuit used in the run-length counter. The square-wave timing-voltage A (Fig. 7) is differentiated by the input capacitor, so that a count is registered at the beginning of the sampling interval when voltage A goes positive.

VI. Conclusions

For the relatively simple, but general classes of stationary random processes studied, the two-level correlation technique appears to be useful in determining the periodicities present in a random process, and characterizing its bandwidth. However, it is not capable of determining the exact amplitude of the correlation function, and it does not provide details about regions of inflection. It is interesting to note that when a two-level random process is uncorrelated at some particular value of τ, statistical independence is also indicated [4].

There are, of course, bandwidth limitations associated with using operational amplifier circuits for performing two-level correlation operations. This would not necessarily be a handicap when applying the technique to simulation studies on a typical analog computer. For other applications, faster circuitry possibly could be developed. The two-level correlator requires a minimum of equipment; it is useful, therefore, in systems where simplicity is important.

Appendix

Derivation of Autocorrelation Functions

Random Noise Input

The random noise source consisted of a random telegraph wave passed through a first-order low-pass filter as shown in Fig. 6. The transfer function of this filter is given by

$$H(j\omega) = e/e_1 = -R_0/R \frac{1}{1 + j\omega R_0 C}$$

where R_0/R is a scale factor which will be taken as unity for purposes of this calculation; therefore

$$H(j\omega)^2 = \frac{1}{1 + \omega^2 R_0^2 C^2} = \frac{1}{1 + (\omega/\omega_c)^2}$$

where

$$\omega_c = 1/R_0 C.$$

The output-power spectral-density $W_2(\omega)$ is related to the input-power spectral-density $W_1(\omega)$ by the following equation:

$$W_2(\omega) = \left| H(j\omega)^2 \right| W_1(\omega)$$

and the input-power spectral-density is given by the Fourier transform

$$W_1(\omega) = \int_{-\infty}^{\infty} R(\tau) e^{-j\omega\tau} d\tau.$$

The input to the filter amplifier is assumed to be a random telegraph wave, with the probability that k traversals occur in a time interval T given by the Poisson distribution

$$P(k, T) = \frac{(NT)^k}{k!} e^{-NT}$$

where N is the average number of zero crossings/sec. The autocorrelation function of a random telegraph wave which assumes the values $+1$ and -1 with equal probability can be shown to be [5]:

$$R_{xx}(\tau) = e^{-2N|\tau|}.$$

Substituting in the expression for W_1 we have

$$W_1(\omega) = \int_{-\infty}^{0} e^{2N\tau} e^{-j\omega\tau} d\tau + \int_{0}^{\infty} e^{-2N\tau} e^{-j\omega\tau} d\tau.$$

After integration this becomes

$$W_1(\omega) = 1/N \frac{1}{1 + \omega^2/4N^2}.$$

The output-power spectral-density is then given by

$$W_2(\omega) = \frac{1}{1 + (\omega/\omega_c)^2} \frac{1}{N} \frac{1}{1 + \omega^2/4N^2}.$$

The output autocorrelation function is found from the inverse Fourier transform

$$R_{yy}(\tau) = (1/2\pi) \int_{-\infty}^{\infty} W_2(\omega) e^{j\omega\tau} d\omega.$$

After substituting for $W_2(\omega)$, we have

$$R_{yy}(\tau) = (1/2\pi) \int_{-\infty}^{\infty} 1/N \frac{1}{1 + (\omega/\omega_c)^2} \frac{1}{1 + \omega^2/4N^2} e^{j\omega\tau} d\omega.$$

This integral can be evaluated by contour integration in the complex plane to yield

$$R_{yy}(\tau) = \frac{2N\omega_c e^{-\omega_c|\tau|} - \omega_c^2 e^{-2N|\tau|}}{4N^2 - \omega_c^2}.$$

For the present case, $\omega_c \ll 2N$, so that after simplification and dropping the scale factor the expression becomes

$$R_{yy}(\tau) = e^{-\omega_c|\tau|}.$$

An expression for the two-level autocorrelation function of a sine wave plus Gaussian noise has been derived by McFadden [2].

$$r(\tau) = 2/\pi[\sin^{-1}R(\tau) + 1/4a^4(1-\rho^2)^{-3/2}(\cos(\omega_s\tau - \rho)) \cdot (2 - \cos\omega_s\tau, -\rho^2) + 0(a^6)]$$

where

$$R(\tau) = \frac{\rho(\tau) + a^2\cos\omega_s\tau}{1 + a^2}$$

and

$\rho(\tau)$ = autocorrelation function of noise,
$\cos\omega_s\tau$ = autocorrelation function of the sine wave,
a = ratio of sine-wave amplitude to rms noise level.

For the case of noise alone, $a = 0$, and

$$R(\tau) = \rho(\tau) = e^{-\omega_c|\tau|}.$$

Therefore, the two-level autocorrelation function becomes

$$r(\tau) = (2/\pi)\sin^{-1}e^{-\omega_c|\tau|}.$$

Sine Wave Plus Random Noise Input

For the case of a sine wave plus random noise, a random telegraph wave and a sine wave were combined in the input to a low-pass filter as shown in Fig. 6. The output voltage for this case is related to the sum of the input voltages by the equation

$$e = H(j\omega)(e_1 + e_2) = -R_0/R\frac{1}{1 + j\omega R_0 C}(e_1 + e_2).$$

Thus the filter transfer function is the same as that previously found for the case of noise alone. The input-power spectral-density is then the sum of that of the noise plus that of the sine wave

$$W_1(\omega) = \frac{1}{N(1 + \omega^2/4N^2)} + K\delta(\omega_s);\quad K = \frac{1}{\sqrt{1 + (\omega_s R_0 C)^2}}.$$

The output-power spectral-density is, therefore,

$$W_2(\omega) = H(j\omega)^2 W_1(\omega) = \frac{1}{1 + \omega^2/\omega_c^2}\frac{1}{N(1 + \omega^2/4N^2)} + K\delta(\omega_s).$$

The output autocorrelation function is again found by taking the inverse Fourier transform of the output power spectral density

$$R_{yy}(\tau) = 1/2\pi N\int_{-\infty}^{\infty}\frac{e^{j\omega\tau}}{(1 + \omega^2/\omega_c^2)(1 + \omega^2/4N^2)}d_\omega + 1/2\pi \cdot \int_{-\infty}^{\infty}\frac{K\delta(\omega_s)e^{j\omega\tau}}{1 + \omega^2/\omega_c^2}d\omega.$$

Integration yields the exact autocorrelation function

$$R_{yy}(\tau) = \frac{2N\omega_c e^{-\omega_c|\tau|} - \omega_c^2 e^{-2N|\tau|}}{4N^2 - \omega_c^2} + \frac{K\omega_c^2\cos\omega_s\tau}{2\pi(\omega_c^2 + \omega_s^2)}.$$

This can be written in simpler form for the present case, since $\omega_c \ll 2N$

$$R_{yy}(\tau) = \frac{\omega_c e^{-\omega_c|\tau|}}{2N} + \frac{K\omega_c^2\cos\omega_s\tau}{2\pi(\omega_c^2 + \omega_s^2)}.$$

This can be shown equivalent to the form

$$R_{yy}(\tau) = \frac{e^{-\omega_c|\tau|} + a^2\cos\omega_s\tau}{1 + a^2}$$

which agrees with the result given by McFadden [2].

Acknowledgment

The authors wish to thank Prof. G. A. Korn, of the Department of Electrical Engineering, University of Arizona, Tucson, Ariz., who originally suggested this project, and whose many suggestions were essential to its completion. Many helpful comments were also provided by Prof. A. P. Sage.

References

[1] J. F. Kaiser and J. B. Angell, "New Techniques in Equipment for Correlation Computation," M.I.T. Servomechanism Lab., Cambridge, Mass., Tech. Memo 7668-TM-2; December, 1957.
[2] J. A. McFadden, "The correlation function of a sine wave plus noise after extreme clipping," IRE TRANS. ON INFORMATION THEORY, vol. IT-2, pp. 82–83; June 1956.
[3] G. A. Korn and H. Koerner, "Function generation with operational amplifiers," *Electronics*, vol. 32, pp. 66–70; November 6, 1959.
[4] H. Cramer, "Mathematical Methods of Statistics," Princeton University Press, Princeton, N. J., pp. 280–283 and pp. 441–445; 1946. See for a discussion of tests for statistical independence by means of the mean-square contingency function.
[5] See for a derivation of the case where the wave assumes values 0 and 1. W. B. Davenport and W. L. Root, "An Introduction to the Theory of Random Signals and Noise," McGraw-Hill Book Co., New York, N. Y., p. 61; 1958.

Soviet Cybernetics and Computer Sciences—1960*

EDWARD A. FEIGENBAUM†

The following is to some degree an updating of information on Soviet computer techniques described in the article edited by Dr. W. H. Ware and published in the March, 1960, issue of these TRANSACTIONS. As was the Ware report, this also is being published in *Communications of the Association for Computing Machinery*, in order to reach the widest possible audience in a timely manner.

—*The Editor*

Summary—**This is the author's report of his visit to the Soviet Union in June and July, 1960. The purpose of the trip was to attend the First Congress of the International Federation of Automatic Control (IFAC) as an official American delegate. The author also arranged to meet with certain scientists in psychology, physiology, and the computer sciences, and to visit some Russian research institutions doing work in these areas. Soviet research in cybernetics, neuro-cybernetics, artificial intelligence, mechanical translation, and automatic programming are discussed, and new developments in Soviet computing machines are described.**

The author describes his discussions with several important Soviet personalities in the computer sciences, which dealt with their particular work and the work of their research institutions. He concludes that Soviet research in the computer sciences lags behind Western developments, but that the gap is neither large nor based on a lack of understanding of fundamental principles. He believes that the Soviets will move ahead rapidly if and when priority, in terms of accessibility to computing machines, is given to their research.

CONTENTS

* Received by the PGEC, September 14, 1961.

† School of Business Administration, University of California, Berkeley, Calif.

I. First International Congress of the International Federation of Automatic Control, Moscow, June 27–July 7, 1960

ONE of the purposes of the author's visit to Moscow was to attend, as an American delegate, the first Congress of the IFAC.

Plan of the Congress

The activities of the Congress were organized in three cities: Moscow, Kiev, and Leningrad.[1] The actual working sessions were held at Moscow State University during the week of June 27–July 2. The sessions in the other cities, July 3–7, consisted only of technical excursions to industries and research institutions.

In Moscow, the Congress scheduled two plenary sessions (opening and closing) and numerous daily technical sessions. Run in parallel with these business sessions was a rather large number of organized technical excursions to various industries and institutes. Also scheduled in parallel were many organized sightseeing excursions of the ordinary tourist variety. An active and highly organized Ladies' Program ran for the full week. The delegate was inundated by a blitz of organized activity.

First Plenary Session

The first plenary session of the Congress, held in the Congress Hall of the University, was heavily attended. The program consisted of a number of welcoming speeches and an address by the well-known scientist and Chairman of the U.S.S.R. National Committee for Automatic Control, V. A. Trapeznikov.[2] The Communist welcoming speeches were heavily loaded with propaganda and exhibited what Western delegates thought to be an embarrassing lack of discretion, taste, and dignity. The Soviet Deputy Premier talked on the problems which automation would bring to "certain societies" which were not well equipped to handle this kind of technological change—change which would bring unemployment, re-education problems, and relocation stresses.

[1] The Russians regarded the choice of Moscow for the First Congress of IFAC as a feather in their caps, and used this opportunity for propaganda effect in the Soviet and foreign press. A special commemorative stamp of the event was issued, special commemorative "first-day cover" envelopes were printed, the opening plenary session was addressed by a Soviet Deputy Premier, and a remarkably sumptuous and ceremonial reception for all delegates and their wives was held on the final day in the Czar's Palace in the Kremlin. Aside from this reception, however, there seemed to be no extra effort made to accommodate these "special guests" of the Soviet Union, and, in fact, the bitterness felt by many of the delegates because of the difficulties and indignities suffered during the conference was not mollified by the splendid Kremlin party.

[2] This speech was a highly significant statement of Soviet attitudes and goals in cybernetics and automatic control. Unfortunately, the speech received little attention and distribution. It was not available in the preprints of the IFAC Congress, and only a few printed English translations were distributed after the session. I am sure, however, that the speech will appear in the final Congress volume when it is published.

Technical Sessions

The technical business of the Congress was divided into four major classifications: theory, components, industrial applications, and general problems. These classifications were further divided into twenty-one categories (listed in Appendix II). Sessions in the various categories were run in parallel, though not every category held sessions every day. The number of papers given during the technical sessions was enormous: nearly 300 papers were presented in the five days of technical sessions. Almost all of these papers were available in preprinted form a few weeks before the Congress. In no instance was I aware of the presentation differing in any way from the preprint.

The presentations were handicapped by translation difficulties. No simultaneous translation was available. Translations were, instead, sentence by sentence and often in more than one language.

The postpresentation discussions in the sessions I attended were generally uninteresting, uninformative, and tedious. Again, severe translation problems marred performance. Tedium was enhanced by the peculiar Russian habit of "discussing" a paper by delivering a 15- to 30-minute extension, clarification, or rebuttal of the paper.

In general, Soviet papers could be characterized as oriented toward theory, while papers of Western delegates mixed theory and application.

Technical Excursions

In conjunction with the Congress, various research institutes, educational institutions, and plants were officially opened to the delegates for technical excursions (listed in Appendix III). The trips were highly organized affairs and consisted of about sixty people each. By far the most popular tour was one to the Institute of Automation and Telemechanics in Moscow. They were canned, planned tours and provided no opportunity for personal contacts or detailed questions. In my opinion they were a severe time-wasting activity, using up the few hours available for personal contacts in Moscow. After speaking with other delegates who took these trips, I am convinced that this judgment is correct.[3]

Person-to-Person Communication

The greatest value of most large conferences lies not

[3] There was an exception. N. Blachman, of Sylvania Electric Products Co., Mountain View, Calif., saw a transistorized digital computer at the Moscow Power Institute. His report on this computer may be found in ONR Tech. Rept. No. ONRL-C-15-60; September 9, 1960, and in the *Communications of the ACM*, vol. 4, pp. 256–265; June, 1961. As far as I know, this is the only American report on this machine. The computer has the following characteristics: speed—25,000 fixed point operations per second, or 5000–7000 floating point operations per second; word length—20 or 40 bits; memory size—4096 words of ferrite core, fixed store of 256 words on condenser-printed paper, magnetic-tape store of 50,000 words; output-numeric only; clock speed—100 kc. Blachman reports that this computer was built by students in three years, beginning in 1957.

in the transaction of formal business, but in the informal exchange of ideas and information among delegates, who should be given ample opportunity to meet and converse.

Interpersonal communication between Western and Soviet delegates was held to a minimum. All foreign delegates were boarded in the enormous Hotel Ukraine on the outskirts of the Moscow central area; Soviet delegates were housed in a number of different hotels in the center of Moscow. It was not possible officially to obtain names of the hotels of Soviet delegates or, knowing the hotel's name, to obtain from the hotel desk a delegate's room number or telephone number. The only list of delegates and their hotel locations which I saw was one printed by the American delegation listing data on American delegates.

There were no official communication channels between delegates, no central communications center for the delegates, no message system, no mailboxes. It was relatively easy to contact any foreign delegate through the desk at the Hotel Ukraine, but it was almost impossible to contact a Russian delegate.

Soviet delegates could be met by seeking them out at the Congress. This was a hit-or-miss procedure with small probability of success. The various parallel sessions were taking place in widely separated areas of the University. Furthermore, the technical sessions were not well attended by Russians, or by foreigners. Most frustrating of all, Soviet delegates did not, in general, wear the big name tags provided for delegates, though virtually every Westerner wore his continuously.

Sidelights on IFAC

In reception and handling, difficulties and indignities suffered by the delegates at the hands of an inept staff were marked. Travel arrangements to Congress cities and to other places were handled with incredible inefficiency.

The delegates arrived in Moscow on Saturday and Sunday. The Soviet passport-checking procedure at the airport broke down in the face of the onslaught of delegates and wives. Transportation arrangements for taking delegates to the Hotel Ukraine were chaotic. At the hotel, delegates were faced with a waiting line of a hundred or more at the Intourist Bureau, where hotel rooms and food coupons were being distributed.

The Congress moved from Moscow to Kiev and Leningrad on July 3. Over 1000 foreigners had to be moved to these and other cities. During the Congress week, inquiries about tickets elicited the following standard reply: "What are you worrying about? It's too early. You're not leaving until Sunday. All tickets will be handed out at noon on Saturday." There were no tickets at noon on Saturday, nor at three, six, nine, or midnight. By mid-evening the Intourist Bureau was crowded with worried, tired delegates queueing up for tickets. After midnight, a list of names and room numbers was drawn up with instructions to Intourist to phone these people when their tickets arrived. Disgusted, I went to bed at about 1:30 A.M., after signing the list. I was not phoned. I understand that others were and that they stood in queues at all hours of the night. I did not get my tickets and thereby missed the 9:00 A.M. flight to Tbilisi. Sunday afternoon, having uncovered the fact that there was a plane to Tbilisi at 6:00 P.M., I made a plea of desperation to one of the Intourist girls to issue me tickets for this flight. With an air of surprise which seemed to say, "How could you get so worked up over such a minor thing?" she issued the tickets and travel coupons in less than five minutes!

Rumors flew all week that the delegates were to be invited to a reception at the Kremlin on Saturday. In typical Russian fashion, no invitations went out and no official announcement was made until the final plenary session of the Congress a few hours before the reception. The party, given by the Academy of Sciences of the Soviet Union, was held on the Kremlin grounds, in the Czar's Palace, a building which is the seat of the Soviet Presidium and is generally closed to visitors. It was a splendid reception, highlighted by culinary luxuries and excellent entertainment—a dizzying performance by our inconsistent Soviet hosts.

Evaluation

In my opinion, the primary value of the Congress was that it provided an opportunity for 135 Americans and many other Westerners to visit the Soviet Union and gain a first-hand impression of Soviet society today. It also gave a limited number of Soviet citizens a chance to meet, talk with, and evaluate these several hundred Westerners. Here I wish to emphasize person-to-person mutual enlightenment, rather than scientific and professional exchange. In these turbulent times of uncertainty and mutual distrust, we cannot have too much of this. If conferences can provide a way of channeling corporate, private, and governmental funds for this important educational endeavor, I believe their worth to be established.

From the viewpoint of mutual exchange of scientific information, the IFAC Congress made a rather poor showing. Most large conferences are bad in this respect, but something useful is salvaged when delegates meet and talk informally. Because of the severe constraints which limited the contact of Westerners with Russians, the IFAC Congress could not claim even this salvage value.

II. The Private Business of one Delegate in Moscow

Problems of the IFAC Delegate

In anticipating the trip I thought that as a delegate to a distinguished conference I would have the kind of

"important guest" status that would open doors and ease my job in making personal contacts. This is not what transpired. The fact is that most IFAC delegates were patently refused admittance, sometimes more subtly than at other times, to those institutions which were not included in the schedule of technical excursions. No amount of taxicab riding, telephoning, insisting, or cajoling could open these closed doors. The number of individual incidents which stand as evidence of this policy is formidable.

My interpretation is that the Russians, faced with the overwhelming problem of controlling the movements of several hundred highly intelligent scientific people, settled on a policy of closing all scientific doors even remotely related to the interests of these people, except those doors which were officially opened as part of the Congress. I suspect that this policy was held in the other two Congress cities as well, for I experienced great difficulty in making personal contacts in Kiev and Leningrad, even with people who were expecting me and even though I arrived in these cities after the other delegates to the Congress had departed.

Examples of my difficulty will come up in the various sections of this report. Perhaps the best example of what was going on in Moscow at the time is one which did not affect me very much. To my knowledge, no delegate gained admittance to Lebedev's Institute of Precise Mechanics and Computing Techniques. Even those who carried written invitations from Lebedev himself, as a result of his trips to the U. S., England, and West Germany, were denied entrance. One persistent English fellow was forcibly turned away at the gate. This is especially surprising since Lebedev's Institute has had more than the usual contact with Western delegations. To sum up, things were bad enough for an American in Moscow in June, 1960; being a delegate to the IFAC Congress only made things worse.

A. P. Yershov

I visited with Dr. A. P. Yershov, Director of Theoretical Programming at the Computing Center of the Academy of Sciences of the U.S.S.R., who is a major force in automatic programming in Russia. His magnum opus, "Programming Program for the BESM Computer," has been published by the Pergamon Press, London, England, in an English edition. Dr. Yershov has moved to Novosibirsk[4] to take up a major post as head of programming research in this new scientific complex.

Dr. Yershov is a man of great intelligence, curiosity, and energy, and he has considerable awareness of the state of the computing art in Russia as well as in the U. S. In Novosibirsk he will have influence and, I believe, relatively great freedom, and, most important, a good modern computing machine (or machines!).

The following notes briefly summarize our talks:

Activity in Novosibirsk: Dr. Yershov's new position in Novosibirsk was a focus of our conversation. An M-20 computer had already been installed in Novosibirsk, and had almost been debugged.[5]

Dr. Yershov was working on an ALGOL-type language, since completed at Novosibirsk, for the M-20. At the time, he had a staff working with him at the Computer Center in Moscow on this project.

In Moscow, programming research people do not influence the design of new machines, which is strongly under the control of Lebedev's Institute of Precise Mechanics and Computer Techniques. In Novosibirsk, however, Dr. Yershov and his colleagues hope to do extensive computer development in which the various "disciplines" in the computer field will cooperate. This cooperation, he insists, is a necessity.

Once established in Novosibirsk, Yershov hopes to begin work as soon as possible on the so-called "logical languages," especially symbol-manipulating list-processing languages whose development, he says, is now essential. He knows of the U. S. list languages called IPL and LISP.

Activity in Moscow: One would expect automatic programming languages to be widely used in Yershov's own shop, but this is not the case. Yershov said that only a very small percentage of all problems done at the Computing Center are run with the aid of a programming program. He gave two reasons: first, and most important, the lack of alphabetic input-output devices, and second, the reluctance of experienced programmers to learn a new system and switch over to its use.

According to Yershov, the Computing Center of Moscow State University is concerned with the solution of practical scientific and engineering problems. No basic computer research is being carried on there, and there is no interest in the theory of programming.

A. A. Lyapunov is, of course, at Moscow State University. Lyapunov is the Norbert Wiener of Russian cybernetics, a world-famous mathematician, and a

[4] Dr. Yershov's new address is: Siberian Division, Institute of Mathematics, Academy of Sciences of the U.S.S.R., Novosibirsk 72, U.S.S.R.

[5] The M-20 is one of the latest of the large Soviet machines. Some data on the M-20 follows: mode—3-address floating point; average number of operations per second—20,000 (Soviet estimate, using statistical mix of instructions based on normal use); additions per second—50,000; multiplications per second—17,000; word size—45 bits; core store—4096 words; complete memory cycle time—6 microseconds; drums—three units with a total of 12,000 words and a transfer rate of 12,000 words per second; tapes, four units, each eight-channel, addressable blocks of 1–4000 words, transfer rate of 2500 words per second; input-output—punched cards and on-like printer; alphanumeric capability—unknown, but probably absent. For information on other Soviet machines, see W. Ware, ed., "Soviet computer technology—1959," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-9, pp. 72–120; March, 1960. Also published as RAND Corp. Research Memo. RM-2541; March 1, 1960.

revered man in Soviet scientific circles. As the patriarch of cybernetics, he helps to organize and transmit knowledge in the field. He is responsible for bringing about various cybernetics conferences in Russia; he founded, and continues to edit, the well-known series, "Problems in Cybernetics";[6] he continues to teach at the University; and he personally supervises the education of a number of promising graduate students. He recently organized a volume of articles on the STRETCH computer translated from U. S. journals and publications.

L. I. Gutenmakher

I spoke with Professor L. I. Gutenmakher, Director of the Laboratory for Electrical Modeling of the Institute of Scientific Information. Despite rumors to the contrary—as far as I can determine—the Laboratory is still part of the Institute, which, I gather, employs about 300–500 people. Gutenmakher himself is a well-known figure both inside and outside the Soviet Union. His early work was with analog computers. In 1949 he published a book entitled "Electrical Models," which dealt with electrical analog models of physical systems. His recent work has included some flamboyant and speculative material on the simulation of human brain functions, and on intelligent information retrieval machines operating with the principles of human associative memory. Gutenmakher's writings on the automation of brain functions are provocative. His ideas on associative memories for computers are, in general terms, quite close to those of the RAND-Carnegie Institute of Technology research effort on simulation of cognitive processes. In 1960 a book of his was published, "Electronic Informational-Logical Machines," which sold very well in Moscow. He is an uncommunicative man and rather unfriendly, and he has a reputation for being a difficult man to get to meet.

We spoke in German about the work of the Laboratory, which is engaged in building a so-called information machine. He referred me to his recent book for more information on the subject. The book, incidentally, gives a "popular" rather than a technical treatment. I inferred that the machine is more of a digital information processor than a digital computer. That is, it is being designed specifically for the problems of information storage, retrieval, and manipulation, rather than for purposes of calculation. When I had questioned Yershov about this machine, he had told me it was an "ordinary" type of digital computer; Gutenmakher contradicted this, claiming that it was not the "ordinary" type but that it had an associative memory. I believe the resolution to this contradiction is that the machine utilizes familiar components in a novel design for information processing. Many of his workers are engaged in constructing this information machine.

Gutenmakher said that he has a large group working on the problems of: 1) information retrieval of chemical data, and 2) mechanical translation of languages. Dr. Yershov had told me that the Laboratory was also working on an "information language." He said that this is *not* a programming language. My impression is that it is a language convenient for coding textual materials and data. The chemical information previously mentioned will be, or already has been, coded in this language. He mentioned that the mechanical translation group is also very interested in using the language, so it is likely that the language has some rather general properties. He mentioned that a brief account of this language is presented in his new book.

I questioned Gutenmakher about the Laboratory. In the course of speaking of its work, he mentioned that there was much to do and little time in which to do all that needed to be done. I asked him how many people worked in the Laboratory. "Enough," was his answer. I asked him for an order of magnitude. "Large," he answered. His researchers (as opposed to laboratory technicians) are mathematicians, logicians, linguists, engineers, and physicists. He said that there were no psychologists and no physiologists. They were not experimenting on human memory, nor were they experimenting in the area of higher mental activity in connection with simulation of brain functions.

In connection with theorem-proving machines, he has some people working on a geometry machine. He did not know of Gelernter's work on a geometry theory machine.

The Laboratory had no game-playing programs.

On the topic of the simulation of mental activities of the brain, his comments were significant. He believes a machine will imitate these activities. I asked him how long he thought this would take: 50 years? 20 years? No, he said, this was too long. Therefore, I told him of the estimates given by Newell and Simon in 1958, that a machine would be a chess champion in ten years, and would discover and prove an interesting new mathematical theorem in the next ten years. Gutenmakher said that this was a much better estimate, but that it was too conservative. He felt that this sort of thing would happen in less time.

A. V. Napalkov

A. V. Napalkov appears on various published papers as an associate of Professor S. N. Braines of the Laboratory for Physiological Research of the Institute of Experimental Psychiatry. He is also on the Faculty of Higher Nervous Activity, Moscow State University.

Napalkov is a physiologist whose basic interest is in machine models of physiological (and psychological)

[6] A. A. Lyapunov, Ed., "Problems in Cybernetics," State Publishing House for Physical and Mathematical Literature, Moscow, U.S.S.R., vols. I–V; 1959–1961.

processes.[7] He teaches at the University, and his work there, in conjunction with Braines' Laboratory, is performed with a staff of young assistants, some of whom are graduate students. His laboratory, which is in the new building of the Biological-Soil Faculty, is modern and impressively well-equipped.

Braines and Napalkov have built a conditioned-reflex learning machine. At the Laboratory they also have a chimpanzee or two, used for complex conditioning experiments. The Laboratory is reported to have carried out the much-publicized experiments with dogs on restoration of youth through long sleep.

Napalkov and I exchanged ideas about brain models. His research method is substantially the same as that of the RAND-Carnegie group: make observations of behavior (of humans and/or animals) in complex environments; construct machine models of the behaving organisms based upon knowledge of what some plausible mechanisms might be; allow the machines to behave in simulated environments; compare the behavior of the machine model with the observed behavior.

Following our discussion, his assistants ran off an experiment in complex conditioning with a dog in which a chain of conditioned responses was interposed in another chain of conditioned responses. The experiment was designed to show that a dog could learn this kind of cycle-within-a-cycle behavior.

Studies of the chaining of responses are typical of the work of Napalkov and Braines. An animal is trained to press a key for some reward. The key-pressing is then reinforced only when a tone is sounded, which occurs only if the animal has previously pressed a certain lever, and so forth. Many variants of the basic experiment have been performed in which: 1) the timing and nature of the chain-stimuli have been varied, and 2) conditions external to the chain (*e.g.*, hunger, thirst) have been varied. In one interesting experiment, two different chains leading to different kinds of rewards (food in one case, water in the other) were taught to rats and dogs. The chains had a common link. The animal was then deprived of, say, food, and subsequently run in the water-chain. The problem under investigation was whether or not the animal would switch at the common link into the other chain (in this case the food chain). Experiments like these provide the behavioral data from which Napalkov, Braines, and their engineers attempt to build machine models of animal behavior.

Napalkov told of having to write many general and "popular" articles about his brain-model research, as well as numerous official memos, in order to convince people with authority that his research should be supported. He said that there was much opposition among physiologists to the machine modeling approach.

Some of the theoretical basis for the Braines-Napalkov learning machine may be found in a small book of working papers written by Braines, Napalkov, and Svechinskiy.[8]

A. R. Luria—The Institute of Psychology

I called upon Professor Luria of the Psychology Department of Moscow State University and the Institute of Defectology. Luria is a distinguished scholar, well known in the West, who speaks perfect English, and has visited and lectured in both England and America. A recent book of his has been published in English.[9]

Luria spoke briefly of his recent work on the influence of language on the initiation and regulation of simple motor and verbal behavior in children. This work is adequately summarized in Pick's report,[7] to which the interested reader is referred.

During our conversation Luria mentioned that Dr. Eugene Sokolov of the Faculty of Higher Nervous Activity at the University was interested in modeling mental processes on a computer.[10]

Luria asked me to talk to the psychologists at the Institute of Psychology about my research.[11] The talk was attended by about fifty people, many of them young. Dr. Smirnov, head of the Institute of Psychology, was chairman of the question period. The session was marked by intelligent questions showing an understanding of my statements about computer models of learning. Dr. Smirnov has himself done research on human memory and has written a book on retroactive inhibition.

After my lecture I had a long talk with a young researcher. It was one of those random meetings which pays dividends. She works at the Institute of Foreign Languages in the Experimental Laboratory for Phonetics, where she does research on speech perception. She is now concerned with the relationship between perceptual processes and memory. She believes that the study of human perception of speech will aid in the solution of some fundamental problems in mechanical translation of languages, *e.g.*, the problems of groupings of words and selection from a set of alternate meanings.

[7] For discussion of Napalkov's work, see: H. Pick, "Some Current Trends in Experimental Psychology in the Soviet Union," Psychology Dept., University of Wisconsin, Madison; 1960. Unpublished manuscript.

[8] S. N. Braines, A. V. Napalkov, and V. B. Svechinskiy, "Scientific Notes (Problems of Neuro-Cybernetics)," State Publishing House, Moscow, U.S.S.R.; 1959. Translated into English by Joint Publications Research Service as JPRS 5880, Office of Technical Services, Dept. of Commerce, Washington 25, D. C.; October 18, 1960. (The book may be purchased for $3.50.)

[9] A. R. Luria, "The Role of Speech in the Regulation of Normal and Abnormal Behavior," Pergamon Press, London, England; 1961.

[10] Some very interesting and informative notes on the work of Sokolov are to be found in the paper by Pick.[7] See also, E. N. Sokolov, "A probability model of perception," *Voprocii Psichologii*, no. 2, 1960. (Available in English translation by Pergamon Press.)

[11] For a discussion of the work of the Institute, see: W. Reitman, "Some Soviet Investigations of Thinking and Problem Solving," Carnegie Institute of Technology, Pittsburgh, Pa.; 1961. (To be published.)

The Laboratory for Phonetics is interested in building a speech-recognition machine, but at present they have no electronics laboratory of their own. Furthermore, they do not have access to a computer. She told me that the Department of Mechanical Translation of the Institute of Foreign Languages has requested a computer, but their request has so far been turned down, and she thinks they will not get the machine.

A Lecture by Norbert Wiener

On June 28 Norbert Wiener lectured to the Russians at the Polytechnic Museum in downtown Moscow. He spoke on the analysis of brain waves and on some results of his electroencephalographic research. Attendance turned out to be standing room only; 500–700 people were present and many others were turned away. Most of the questions were concerned with the comparison between brains and machines. It was obviously a topic of much concern to the Russian scientists. The most vigorous applause of the evening came when, in answer to a direct question, Wiener stated his belief that the creativity of man would always find a higher level than the creativity of a machine. The Russians were clearly gratified by this answer.

III. The Business of one Tourist in Tbilisi, Capital of the Georgian Republic

Computing Center, Academy of Sciences of the Georgian Republic

In contrast to my experiences in Moscow, I had no trouble with my appointments in Tbilisi. One call to the Computing Center sufficed to bring someone quickly over to my hotel. Professor Kueselava, who is Professor of Mathematics and Director of the Computing Center of the Academy of Sciences of the Georgian Republic, and part of his staff, greeted me on arrival. He said that he had received my letter and was glad I was able to come.

The Center is currently housed in a poor building in an old, but central, section of Tbilisi. A new building is currently under construction but is in a more outlying area. The Center in Tbilisi is a relatively recent phenomenon. It was opened in 1958 and staffed by mathematicians, scientists, and engineers drawn from the local University and Polytechnic Institute. These people were trained in computing in Moscow (two to three years of training). The Center now employs two hundred people, many of whom are technicians and engineers.

The Center operates a URAL computer (serial No. 78), which is installed in a rather sloppy fashion in a large undignified room in the half-finished new building. Down the hall in an unfinished room the BESM II was being assembled. The various functional units (adder, memory, frame, etc.) were sent down individually from Moscow and assembled into a machine by the Center's Moscow-trained engineers. They estimated that the machine would be completed by about January or February, 1961. As far as they know, there is no factory that turns out complete BESM II machines. They are probably right. In Latvia a BESM II is currently being installed in just the same way as the machine in Tbilisi.

The Center is young and still feeling its way, trying to become established. As effective computer installations go, this Center is still in a rather primitive stage. Their URAL computer is slow: its output is on the narrow tape of a 10 (or possibly 14) digit per line "adding machine;" its input was by punched 35-mm film. I was peppered with questions about the reliability of U. S. computers. The implication was that the URAL had low reliability.

Concerning programming, the Center works mostly on practical problems, but some research in automatic programming and mechanical translation has been undertaken. At the time, Center personnel were engaged in writing special subroutines for URAL (and, I presume, BESM II) in connection with problems in physics. In fact, their main interest at the moment is with the solution of physics problems, and they work closely with the Research Institute of Physics on these problems.

The mathematicians at the Center are working on boundary value problems, eigenvalue problems, evaluation of integrals, and solution of differential equations. They told me that they have developed a programming program modeled after Yershov's program. I told them I was interested in understanding this system and would like to write a program in it. They wanted me, instead, to write my program in URAL machine code, but I declined. However, it developed that their programming program existed on paper only and was not really a working compiler. The reasons for not developing it further were that: 1) URAL is too small and slow to accept a decent compiler, and 2) experienced programmers would not use it. Sometime in 1961 they expect to begin their machine translation program in earnest, and indeed, some of the staff is already working on it. The Institute for Automatics and Telemechanics in Tbilisi is already working on translations of Russian into Georgian, and the people in the Center hope to do some joint work with the Institute.

These people are isolated from the computer world of the West. My impression was that they do not know of Western computer journals, automatic programming efforts, artificial intelligence studies, etc., and that they have only the sketchiest information about Western machines. Many of their questions were about Western computers. For example, they asked about the characteristics of the machine that plays chess (*i.e.*, the Newell-Shaw-Simon chess program). In answer, I reported the general characteristics of JOHNNIAC (an

antediluvian machine by 1960 American standards). They were particularly impressed when I referred to JOHNNIAC as an "old machine." They plainly did not believe me when told the memory size of some U. S. commercial computers.

The Computing Center at Tbilisi services the Georgian Republic. They told me that industry does not have its own machines, nor does the University. All problems are brought to the Computing Center of the Academy of Sciences, where they are handled on a closed-shop basis.

One of the most interesting conversations I had at the Center concerned the nature of control of research activity in the U. S. We first discussed the chess machine.

"What is the practical use of such a machine?" they asked.

I replied that there was no immediate practical use for such a machine; that the people who undertook this project wished to study the nature of the problem-solving process in human beings and for computers.

"But who allows you to do this?" they asked. "Who tells you that you can do such research?"

In my reply I tried to explain the nature of research done at universities—that problems are undertaken by faculty members because these faculty members are stimulated by the topics, not because the research is necessarily oriented toward any kind of practical solution of a problem.

"But who allows you to do it?" they continued. "Who gives you the money? Where do you get the computers?"

I explained that the universities sponsored some research and that, in some cases, universities had computers of their own which they made available for faculty research.

I also explained to them the "foundation" system for sponsoring research. I told them about Ford, National Science, and other foundations, and explained how people who want to do research get grants from these foundations.

They were amazed and very impressed. They had no idea, they said, that such institutions existed, and they thought the system was a very good one.

In previous conversations I had told them of the geometry machine constructed by Gelernter at the IBM Research Center. They questioned me further about this. They understood, they said, that American companies were profit-motivated. How was it, then, that a big American company would sponsor such impractical research as a geometry-proving machine? Was not research on thinking machines and learning machines impractical? I tried to explain to them that, although the big companies are in fact motivated by profit, basic research is good business because out of this year's basic research spring next year's good ideas. They nodded their heads. Yes, this was true. They understood.

They also asked me about the National Physical Laboratory in England where I spent the 1959–1960 academic year on a Fulbright Grant. I told them about government-sponsored research laboratories in America and England. They were impressed, and said they did not realize there were such government-sponsored institutions in England and America.

It was obvious to me that these people were highly constrained in their choice of research problems, selecting only those which were important and practical, rather than those which were merely interesting from the point of view of their own curiosity. If they worked on advanced problems of computer applications, they did so because the signal had come from Moscow that they could or should work on these problems. This provided a most interesting insight for me on how things get done in Soviet science.

Institute of Electronics, Automatics, and Telemechanics

I had not planned to visit the Institute of Electronics, Automatics, and Telemechanics of the Georgian Academy of Sciences, yet I encountered no difficulty in arranging a visit at a moment's notice. The Institute is within short walking distance of the Hotel Intourist. It is three years old and employs 200 people: engineers, physicists, mathematicians, linguists, and technicians. There are five divisions: Electronics, Automatics, Telemachanics, Control of Industrial Processes, and Mechanical Translation of Languages. The Division for the Control of Industrial Processes is the largest. Examples of its work include development of certain tooling processes with lathes and control of industrial metallurgical processes (*e.g.*, the rolling of steel).

The division I was most interested in was Mechanical Translation, the smallest of the groups, but I was not able to contact anyone from this group. I do know, however, that there are six people working on Russian to Georgian translation algorithms at the Institute and that, in addition, there are some mathematical linguists working on studies in logical and statistical analysis of Russian and Georgian as an aid to the translation work.

The Electronics Division is currently doing experimental work with digital equipment using vacuum-tube technology, and they are beginning some work with transistors.

The Automatics Division is largely concerned with analog machines for regulation in specific practical applications (an example cited to me was autopilots). They seemed to be little interested in the theory of controllers, theory of automata, or general cybernetic theory. However, a paper describing research on control using homeostat-like devices was presented at the IFAC Congress by the Vice Director of the Institute, Chichinadzye, and is available in the Congress preprints.

A point to note, and one of possible interest to students of Soviet scientific organization, is the apparent duplication of effort on mechanical translation in Tbilisi, a relatively small and provincial outpost of science. Some collaboration of effort is expected, but at present this is minimal. Why, then, are there two separate research teams within blocks of each other? Is this a reflection of a "decentralizing" philosophy in planning basic research in the computer area? Or is it an accident resulting from peculiar decision-making channels in a complex and confused bureaucracy?

IV. Trials of an Unwanted Guest in Yerevan, Capital of the Armenian Republic

S. N. Mergelyan

Academician S. N. Mergelyan is Director of the Scientific Research Institute of Mathematical Machines of the Armenian Academy of Sciences in Yerevan. He was a member of the Russian computer delegation that came to the U. S. in 1959. Mergelyan's Institute (as it is called in Yerevan) was then developing three new compupters: ARAGATS, RAZDAN, and YEREVAN, which have subsequently been completed.

My attempts to visit Mergelyan's Institute were frustrating, and they metwith eventual disappointment. After many telephone calls over a two-day period, I was told that I should stop bothering the Institute with my telephone calls. They did not want to see me, and I could not come over, they said.

M. G. Zaslavskiy

I spoke with M. G. Zaslavskiy of the Physical-Mathematical Faculty of the University at Yerevan. Though he is on the faculty at the University, Zaslavskiy spends part of this time at the Computing Center. He is a mathematician who divides his effort betwen two kinds of studies: 1) theory of functions and theory of algorithms, and 2) theory of finite automata. He said his goal was to develop a general theory of algorithms at the foundations of mathematics, of which algorithms in the form of computer programs will turn out to be special cases. In connection with this goal he wishes to study the link between the theory of algorithms and a general theory of computer programming.

Some of his recent work has been in the theory of constructive functions. He has also been engaged in research on algorithms to define the addition and multiplication operations in number systems more suitable to the representation of real numbers than present systems. All of this work is at the moment highly theoretical.

In the theory of automata, Zaslavskiy has been studying the kinds of descriptions which can be given to events, and how such descriptions can be represented in automata having various numbers of states. He ties this work to the work of Turing, Moore, and Glushkov. He stated that this work has just begun, and there are at present no results.

Computing Center of the Armenian Academy of Sciences

The Computing Center of the Armenian Academy of Sciences is physically located in a relatively new building in Yerevan. I learned from a taxi driver that the building was two or three years old. Later, Center personnel told me that it was only six months old.

In my visit, I was led directly to the office of the Director of the Computing Center. The Director, Alexandreyan, was not present, and his assistant, Tsaakian, presided over the interview. The only other person present was B. M. Grigorian, head of the mechanical translation project at the Center, who spoke very good English.

The Center is organized into three divisions: 1) Theoretical Explorations, 2) Programming, and 3) Machine Translation of Languages.

The Division of Theoretical Explorations is currently working on problems of differential equations, functional analysis, theory of algorithms, and problems in mathematical logic. The head of the Computing Center is also the head of this division. Zaslavskiy is affiliated with this division.

The Division of Programming is concerned with the solution of practical programming problems. It seems that this Division is not working on any advanced techniques, nor is it constructing any kind of automatic programming language.

The Division of Mechanical Translation, headed by Grigorian, is presently working on the problem of translating the Armenian language into an intermediate language, an artificial logical language. Grigorian, a philologist by training, also lectures at the University. The Division consists of twelve people who work on a permanent basis, plus some interested people from the University; it is gradually being enlarged. When I asked Grigorian about the interest among philologists in mechanical translation, he told me that the course he offered in mechanical translation at the University in 1959 was attended by about forty or fifty people. This figure might not be surprising for a university in Moscow, but in Yerevan it is astounding.

I judge that the Center's program in mechanical translation is not far along. Under their translation scheme, a sentence is analyzed at three levels: 1) graphical analysis of symbols, 2) morphological analysis, and 3) syntactical analysis. The end product of translation of a sentence from Armenian into the intermediate language is a string of numbers representing the words and their interconnections. After a word is located in the interlingual dictionary, its number is entered into the translation. If there are various meanings for a

word, the conjunction of all the various meaning numbers is entered. So far, the resolution of multiple meanings is accomplished only by the use of grammatical clues.

Grigorian said he was writing an automatic programming language for mechanical translation. When I probed on this point, it turned out that the automatic programming language was really just a convenient set of subroutines for analyzing various features of sentences. Grigorian called these subroutines "operators." In the language there is a total of fourteen operators: some of these search for various features of sentences; one reduces disjunctions; another is for output; and so on. As far as I could determine, this language embodies no new ideas about memory structures for information processing.

In 1960 the Center had only one computer, an M-3. I was told that they consider this machine much too small for their present purposes. An M-20 is on order from Moscow and should be in Yerevan by the end of the year. Also on order, locally from Megelyan's Institute, is an ARAGATS computer, but the date of arrival of this computer was uncertain.

V. Hasty Business in Kiev, Capital of the Ukrainian Republic

Background

Kiev is touted as a major center for cybernetic research in the Soviet Union. Mentioned in various publications were the following research projects which were supposedly going on in Kiev:

Simulation of the higher nervous system.
Pattern recognition.
The mathematical foundations of constructing machines for diagnosis of heart ailments.
Machine learning and nerve synthesis.
Programs for checking the validity of mathematical proofs.
Automatic machine make-up of train schedules.
So-called "economic cybernetics."

A cybernetics conference in Kiev in 1958 had set up an Institute of Higher Nervous Activity in Kiev under the direction of Professor Amosov. While in London I made the acquaintance of a traveling Russian by the name of Dr. Ivakhnenko, whose work in Kiev was in the field of "technical cybernetics," the theory of self-optimizing control systems. I was, therefore, most interested in exploring the status of research in these areas in Kiev.

In a scheduling decision which in retrospect I consider ill advised, I had planned a visit of only two days in Kiev. I arrived a day late in Kiev because of air-travel difficulties. There had been some flexibility in my schedule which I attempted to exchange for two extra days in Kiev. Unfortunately, Intourist would not allow it, and the whole of my Kiev activity was crowded into one day.

Computing Center, Ukrainian Academy of Sciences

My call to the Computing Center of the Ukrainian Academy of Sciences brought a quick response, and I was immediately taken there by car. The Center is located in a relatively new building in an outlying district of Kiev. I was greeted with warmth and enthusiasm by Dr. Glushkov, Director of the Center, and a top theoretician. He spoke in some detail about his work. He spoke English haltingly, but seemed to understand everything I said.

The Computing Center has a so-called KIEV computer which was built by the Center, but we spoke very little about it. We talked mainly about the research work of the Center and the work of Glushkov himself. He is concerned primarily with problems of abstract automata. However, he has also been working (with graduate students) on the problem of teaching a machine to recognize a meaningful phrase.

A limited vocabulary, consisting of twenty nouns, fifteen verbs, and fifteen to twenty adverbs, is selected. Glushkov pointed out that once these words are chosen, it is possible to enumerate all sentences containing these words and then tell the machine which of these possible sentences are in fact meaningful. But because of the number of possible combinations, this is obviously the wrong way to approach the problem.

The goal is to expose some meaningful phrases to the machine, and, after certain processing, have the machine tell you whether or not a new phrase is meaningful. If the machine makes a mistake, it is told that the response was in error. The idea is to have the machine form classes from particular instances given to it. For example: Suppose a class of "standing objects" is set up—house, boy, man, child. When sentences with the verb "think" are given for the first time, the machine will correctly announce that "boy thinks," "man thinks," and "child thinks" are meaningful phrases, but will incorrectly announce that "house thinks" is a meaningful phrase. The machine will then start a new class for the variant item, that is, a class of objects which "stand but do not think." Admittedly, the proliferation of such classes may be quite large, but, Glushkov maintains, much smaller than the set of sentences arrived at by enumeration. This class-splitting and class-building process is also carried out for noun-verb-adverb sentences.

Another active research project at the Kiev Computing Center is pattern and character recognition. This research is accurately described in the report of the 1959 Computer Delegation.[5] The work has proceeded in two phases. The first phase, which is now complete, consists of a tracking program which scans letters

and follows their edges. The essential principle is elegantly simple. A cathode-ray beam provides a light spot which is focused through a lens onto the scanned letter. The light reflected from the letter is collected by a photo-multiplier tube. The collected signal gives light-dark information. The beam moves with unit steps in small squares, clockwise in a white field and counterclockwise in a black field. This process is illustrated in Fig. 1.

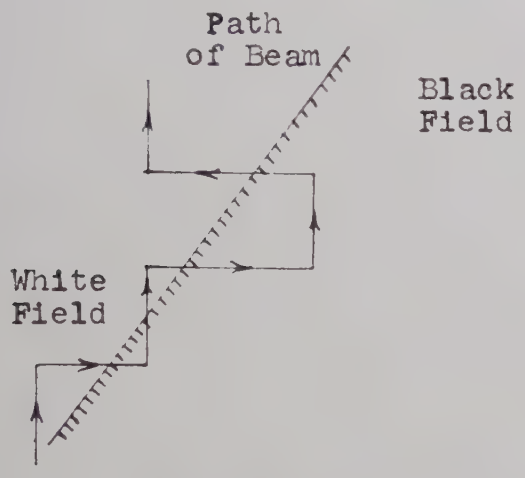


Fig. 1—Movement of beam in character recognition of Kiev Computing Center.

To prevent cycling of the beam in an all-white or all-black area, the tracking mechanism adjusts itself to move two units at a time when two successive unit moves have produced no change from white to black or black to white. This simple mechanism will result in a rough tracking of the edges of a letter.

At each point where the beam changes direction, a new average direction is computed and coded. In this coding, only eight directions are recognized: 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°. A recognition system based on this coding is, therefore, insensitive to slight rotational changes in the letters. Large rotations, however, cannot be tolerated. It was mentioned that this lack of discrimination is not a source of concern because "rotation of letters is never an allowable operation in ordinary printing and reading."

The second phase of the project is as yet unsolved. It concerns the problem of what to do with the codes that result from tracking a letter. Currently, the people at the Computing Center are gaining experience with the kinds of codes produced by the tracking system operating on real letters. They hope to find certain invariances among the codes taken from the same letter. What they are looking for is a set of good discriminators which will sort out the various letters, digits, and characters. This effort as yet has not been successful, and they suggested that they need a great deal more experimentation with the system.

Concluding Notes

The one day available to me in Kiev was spent entirely with Glushkov. Since I had no time to explore my original hypotheses about research activity in Kiev, I was forced to obtain information second hand from my host and his colleagues at the Computing Center.

1) There is little contact between the Institute of Higher Nervous Activity (Professor Amosov, Director) and the Computing Center research group. Glushkov knew of no activity at the Institute on the construction of mechanical brain models, brain model simulation, nerve-net synthesis, or other theoretical or experimental work on brain functions (using either special purpose machines or computer programs). The extent of Glushkov's knowledge was that the Institute has been doing some work in the area of medical diagnosis by computer.[12]

I find it difficult to believe that this is the actual state of affairs at Amosov's Institute. The Institute was set up by special fiat at a cybernetics conference in Kiev in 1958, to foster cybernetic research in Kiev. The title of the Institute is indicative of the intentions of the Academy of Science in forming the group. However, no one in Moscow knew much about the group, which may indicate that it is indeed moving slowly.

2) The Computing Center itself has no project in medical diagnosis by computer and is planning no such effort.

3) Glushkov knew of no work on nerve-net synthesis, learning machines, or thinking machines in Kiev.

4) Of the much-touted work on the so-called economic cybernetics (*e.g.*, large-scale economic planning by machine), there is no work under way presently at the Center. However, Glushkov expressed the hope that a project in this area would soon develop.

Machine economic planning is a research area of tremendous economic importance to the Soviet Union. Significant Russian progress in this area will increase Soviet economic potential. A criticism directed against the Soviet centralized (or relatively centralized) system for economic allocation of resources is that it is impossible to plan or coordinate centrally for an economy on as large a scale as the Soviet Union. The advent of sophisticated ways of using computers could change this. Large-scale, centralized, efficient economic planning and control by digital computers may be feasible in the relatively near future. Soviet progress in this area is of great interest.

5) On the applications of computers to operations research problems, the Center hopes to begin work soon in the area of linear programming. Specifically with regard to automatic construction of schedules, railway

[12] H. Pick of the University of Wisconsin, Madison, in a private conversation, suggested that Amosov's Institute was studying thinking processes of individuals with brain damage.

timetables, and similar problems, Glushkov knew of no such research project in Kiev.

I was not able to visit either the Institute for Electrical Engineering or the Institute for Automatics of Gosplan. However, from conversations with various American delegates, I understand that the directors of these institutes, Milach of the Institute for Electrical Engineering, and Melnik of the Institute for Automatics of Gosplan, attempted to meet reasonable requests for visits.

VI. Brief Notes from Leningrad

In preparation for my visit to Leningrad, I wrote to Professor Vallender of the Computing Center of Leningrad University and Professor Kantorovich of the University. A month before I came to the city, Professor W. Reitman laid a foundation for other interesting contacts; one of these was the well-known and highly regarded physiologist, Mme. Chistovich. In Moscow, scientist friends had suggested other people and places: Dr. Andreyev, of mechanical translation fame, and LOMI, the Leningrad Division of the Steklov Mathematical Institute of the Academy of Sciences of the U.S.S.R. One of the projects of the Institute is the development of PRORAB languages, with which Kantorovich's name has been associated. I had many potential contacts in Leningrad and anticipated a heavy schedule of activity.

Although the midnight sun shone in Leningrad, the climate for cross-cultural contact was frigid. I was not the only American scientist in Leningrad at the time to experience this phenomenon. Other scientists, delegates of the IFAC Congress, at least one of whom gave lectures in Leningrad during his visit, had equally difficult dealings. It was a frustrating few days. People were either out of town, unavailable, ill, or busy. Intourist made contacts difficult: when I asked Intourist to contact a Mr. Karimov for me (a lawyer interested in cybernetics), I was asked why a cybernetics scientist would want to speak with a lawyer.

LOMI

I had the address of LOMI; I took a taxi and arrived unannounced. They were indeed surprised. They held a quick conference and sent me back to the hotel, ostensibly for an interpreter. This was, in fact, a delay, for when I returned with the interpreter, she was of little use. She could not translate technical terms; the people at the Institute spoke fairly good English; and besides, they were not in the mood to communicate.

LOMI is located in an old dilapidated building which it shares with another institute (which I believe is the Institute of Electromechanics). From what I could gather, LOMI has a STRELA computer, but I did not see it. They asked me to define carefully what I was interested in discussing. I mentioned nonalgebraic computer languages, PRORAB languages, and the development of PRORAB languages at LOMI. Thus began one of the most noncommunicative briefings I have ever experienced. Interspersed between comments of some information content was cross-chatter in Russian about what they could and could not tell me.

It seems that there are currently five classes of PRORAB languages: 1) Matrix PRORAB, 2) Polynomial PRORAB, 3) Algebraic Compiler PRORAB, 4) Universal PRORAB (or, as they later called it, Program PRORAB), and 5) Algebraic Transformation PRORAB. The only one of these we discussed was Polynomial PRORAB; I asked about Universal PRORAB, but received no answers.

Polynomial PRORAB is a nonarithmetic analytic computer language for operating on polynomial expressions. Some of the operations which they mentioned as included in the language are truly remarkable for the current state of the art of symbol-manipulating languages. For example: differentiate a polynomial; integrate a polynomial; and solve a polynomial equation. It is difficult to believe, in fact, that they have what they say. The STRELA memory has only 2048 memory cells. The PRORAB system for polynomials, they claim, occupies only 350 cells and the rest are available for working memory. They were annoyed when I expressed incredulity. I asked about the size of the programs which realized the individual operators. The differentiation operator, they claimed, was programmed with only fourteen instructions. They showed me a sample of PRORAB programming, which was impossible to decipher, for it was coded entirely with numbers (they have no alphabetic devices).

Other Individuals

Gerschuni, whom I did not meet personally, of the Physiological Institute (Academy of Sciences, U.S.S.R.) in Leningrad was mentioned as having an interest in sensory processes and information theory.[13]

Professor Kantorovich of the Leningrad University probably will not be located in Leningrad in the near future. He will be moving to Novosibirsk where he will direct work in mathematical economics. Mme. Chistovich, the physiologist, will also be moving; she has been called to Moscow to the Academy of Sciences.

VII. Intelligent Conversations on Intelligent Machines in Riga, Capital of the Latvian Republic

The program of a cybernetics conference held a few years ago in Kiev lists a paper on "Self-Instructing Electronic Computers," by E. I. Arin. I learned, while in Moscow, that Arin was Director of the Academy of Sciences Computing Center in Riga, Latvia.

[13] Dr. W. Ross Adey of UCLA in recent private conversations about his visit to Leningrad, confirmed this report.

Dr. E. I. Arin

Immediately after checking in at the hotel in Riga I phoned Arin. Much to my amazement, I was able to reach him with no trouble and a meeting was arranged in short order. In our conversations we were joined by a young scientist, speaking good English, who was directing the new mechanical translation project in Riga.

Until recently, Arin was Director of the Computing Center of the Latvian Academy of Sciences, which had only a small M-3 computer. Now, however, the Computing Center at the Academy has been de-emphasized, and a major effort will be made to build a strong Computing Center at the Latvian State University at Riga. Arin was appointed Director of this Center and for the past year has been working at the administrative task of organizing the Center and obtaining a strong staff. Thus, his own research has come temporarily to a halt.

Arin mentioned that the focus of computing-machine activity was shifted from the academy to the university, for it was felt that education in this area was a high-priority goal. There is a prime need to train students, and it was felt that the Computing Center (in a city which could have only one) should be at the University. Arin estimated that by the end of the year the University would get its BESM II computer. Following the Soviet style, some of the pieces (the adder and the frame) had arrived from Moscow, and Arin's engineers were assembling the machine. Arin wants to start the Center with a nucleus of about fifty people; he is making trips to other cities in the U.S.S.R. to try to induce well-known people to work with him in Riga. Incidentally, Riga, with its beach and generally favorable climate, is often mentioned by Russians as a beauty spot and a highly desirable place to visit.

Arin's plans for the Computing Center include: 1) a strong group in differential equations and numerical methods for computers, and 2) a more theoretical section working on intelligence in machines and other advanced topics. It is interesting that he plans to build this second section mainly from his university students. His reason, he indicated, was that he had no other resources to tap in the field of intelligent machines in the other cities of the Soviet Union. He would have to train students.

The typical situation will prevail concerning the utilization of the Computing Center. The BSEM II will be the only big computer in Latvia. The Center will handle the problems of the University and the problems of industry, both of which will have priority over basic research on computers. There will, of course, be an initial period of low utilization, during which Arin and his students hope to get some research done. After that, most of the computing time will be occupied by higher-priority problems, and Arin is pessimistic about the chances of doing research on this machine. Arin mentioned that the high-priority computing projects of the Center were problems in quantum physics and crystallography.

Arin's Work on Intelligent Machines

We discussed Arin's so-called self-instructing electronic computer. The work was accomplished in 1955 and 1956 at the Academy of Sciences Computing Center in Moscow; Dr. Yershov was one of his good friends there and had helped him carry out this project.

Arin began the explanation of his problem-solving machine by reviewing an experiment of Pavlov.[14] In the experiment a monkey is taught the solution to a complex problem by first being taught the solutions to a series of simpler subproblems. Each new subproblem is built up from the previous problem by adding a new element to the task. The monkey is taught to pick up and eat an orange. The orange is then encircled by a ring of fire. The monkey is provided with a cup of water. He learns to use the water to put out the fire and then pick up the orange and eat it. The water is then made inaccessible. The monkey is given a jug; he learns to retrieve water to put out the fire; etc. Arin was motivated by the experiment to program this kind of learning mechanism for a computer to enable the machine to learn to solve complex problems.

The program learns to solve simple algebraic equations of three types:

Type 1) $ax+bx+c=dx+ex+f$. The form of the answer is $x=f'$.

Type 2) Two simultaneous equations in two variables, $ax+by=c$, $dx+ey=f$. The form of the answer is $x=c'$, $y=f'$.

Type 3) Two simultaneous equations in three variables, $ax+by+cz=0$, $dx+ey+fz=0$. The form of the answer is $y=f_1(x)$, $z=f_2(x)$.

Only six parameters were used in all of the problems. Six memory cells held the values of the parameters a, b, c, d, e, and f, which were initially set to zero, and a result form (or result vector) for each type of problem was stored. For example: for type 1 problems the result vector was $(1, 0, 0, 0, 0, f')$; for type 2 problems the result vector was $(1, 0, c', 0, 1, f')$.

The rules of algebra were incorporated in the program as allowable transformations of the coefficients. For example, rule 1, a combining operation, transformed coefficient vector (a, b, c, d, e, f) into vector $(a+b, 0, c, d, e, f)$. Rule 2, multiplication by a constant k, transformed the coefficient vector into (ka, kb, kc, kd, ke, kf). Rule 3, multiplication of one equation by a constant k, transformed the coefficient vector into (ka, kb, kc, d, e, f). Naturally, not all rules are applicable to all problem types, but the machine keeps a list which

[14] A description of Pavlov's experiment may be found in: I. A. Poletaev, "Signal: O Nekotorykb Poniatiakh Kibernetiki," Sovetskoe Radio, Moscow, U.S.S.R.; 1958.

associates problem types, result forms, and the list of rules applicable to problem types.

A restricted domain of integers is used: the set $z=[-50, -49 \cdots +50]$ but not including $-1, 0, +1$.

The experimenter poses a problem of type 1, 2, or 3 to the program. The values in the six coefficient working cells are set to zero. A coefficient a, b, c, d, e, or f is chosen randomly. The value for this coefficient is then chosen randomly from z. Thus, for example in a type 1 problem, this step might give a vector (0, 7, 0, 0, 0, 0) corresponding to the equation $0x+7x+0=0x+0x+0$; that is, $7x=0$. Rules are now selected in an attempt to reduce this form to the form of the result for the problem type. The selection is performed randomly, but the rules have attached to them probability weightings of selection which vary with the experience of the program and which reflect the usefulness of particular rules in aiding problem solution. Hence, useful rules eventually achieve a high probability of selection. For any one event, a rule is judged useful if its application meets one of two criteria—if a coefficient is reduced to either 0 or 1. When the result form is achieved, the program which achieved the success—that is, the problem in which the sequence of rules applied—is stored.

The entire problem is now repeated five times using this program with other randomly chosen coefficients. This is done as a test of the generality of the results. If success is achieved on all of the repeated trials using the program, then the program is considered a success and the machine continues its learning.

The machine then adds another coefficient, chosen randomly from the set z, to the coefficient vector. For example, the vector might become (0, 23, 0, 0, 0, 16). An attempt is made to find a set of additional program steps which will transform the new problem (slightly more complicated than the old one) into the form which the last program of rules was able to solve. The result is another program, this time slightly more complex. The process iterates until a program of rules is achieved which will solve problems of this problem type which have all six coefficients.

The machine language program which realizes this consists of 12,000 STRELA commands. The STRELA operates at 2000 operations per second, and requires about 40 minutes to solve one problem with this procedure—that is, to work out the program of rules for solution of a particular problem type.

Any one of the problems on which the machine works as part of the "chain of problems" leading to the ultimate program for solving the problem with six coefficients is of this form: transform some equation form (specified by an "existence vector" for various coefficients, *e.g.*, 1, 1, 0, 1, 1, 1) into the result form. When the machine discovers a program, it stores this program with a "key" that signals that this program transforms some "coefficient existence vector" v into the result vector. At each successive stage of complication of the problem (*i.e.*, the building up of bigger programs) the machine searches for a rule which will transform the equation form with which it is working into a form for which a program already exists. For problem type 1, this can always be done with the addition of one new program step at each complication of the problem. For problem types 2 and 3, more than one additional program step is needed in general, but Arin has proved that no more than three are ever needed.

Suppose that more than one program step is needed. How is such a linked set of program steps arrived at? A single program step is attempted (say, one involving the multiplicative constant k). If in two hundred trials with various k's this step does not produce a transformed form for which a program already exists, then it is assumed that this rule is no good, and another rule is selected. When all the "one-step" tries are exhausted, random combinations of "two steps" are tried, and so on.

When I observed that this was a very inefficient procedure, Arin agreed, but said that in practice it wasn't so bad because of the action of the probability-weighting procedure for selecting rules. What I understood least well in Arin's presentation was the procedure used for choosing k in the application of the multiplicative rules, say rules 2 and 3. Arin observed that choosing k randomly from combinations of elements of z was much too "unlikely" a procedure. Instead, k is computed in one of three forms:

$$\text{a)}\ k = k'/k'',$$
$$\text{b)}\ k = k' \cdot k'',$$
$$\text{c)}\ k = k' + k'',$$

where k' is either a member of the set $u[-1, +1]$ or $v[z]$, and k'' is either a member of the set $u[a, b, c, d, e, f]$ or $v[z]$. For any particular trial k is chosen in the following manner: 1) A form for k is randomly chosen (form a, b, or c above). 2) The nset u or the set v (in the definition of k' and k'') is selected randomly. 3) Particular elements in u or v (whichever is selected) are chosen randomly, giving a particular k' and a particular $k.''$ 4) k is computed. Initially, all the various alternatives in the random choices (which comprise the decision just described) are given equal probability. As the system accumulates experience, those alternatives which prove to be useful have their probabilities of selection incremented at the expense of the unsuccessful alternatives. Hence, finding the right kind of k becomes easier, in general, with the experience of the program.

Arin added one brief note. The process which selects k uses choice by "analogy." An example of the analogy heuristic is the following: if $1/b$ was a useful k in reducing one of the coefficients (b) to 1, then try other combinations of this form, say $1/a$, or $1/c$, to reduce other coefficients. We also discussed schemes for providing the program with examples and allowing it to discover the rules of algebra for itself, but this has not yet been

done. Our conversation then moved on to a more general discussion of problem-solving machines.

Arin thinks of his algebra-learning program as "a toy," an experiment devoid of useful purpose. He is unhappy with it because it is "too blind, too stupid." He clarified this to mean that it operates with too much randomness. It was designed in this way initially because he did not want to give the machine too much information about the task (remember, he took his cue from Pavlov's problem-solving monkey). Arin was particularly dissatisfied because his algebra machine "worked forward"; he felt that "working backwards" was necessary for effective problem solving. This statement led into our discussion of problem-solving heuristics.

Arin displayed a deep understanding of the research problem at the core of studies of intelligent machines. He diagrammed a problem-solving task for me as a labyrinth of possible avenues toward a solution. He said that measures of progress toward the goal were essential for the solution of a problem. In problems of numerical approximation, the notion of progress was obvious. In intelligent problem solving, however, progress was much less clear. He mentioned the use of cues to constrain exploration of the labyrinth. He said that cues could come from the problem information itself, or from outside the problem, though he was not sure how one brings external cues to bear. To illustrate he offered this example: suppose one knew that the goal-exit of the labyrinth was near an exterior pool of water. From inside the labyrinth one could throw out stones and listen for a splash. The sound of the splash could then direct the search toward the exist.

As Arin spoke on this topic, I saw that many of the basic pieces of the heuristic theory of problem solving developed by Newell, Shaw, and Simon were present in his thinking. Interestingly enough, he seemed not to have thought of the role that experimental psychology might play in his research on intelligent machines, even though his work on the algebra machine was triggered by an experiment of Pavlov's.

At one point in our discussion, I mentioned the Newell-Shaw-Simon chess-playing program. Unlike others I met who dismissed the program as useless or impractical, Arin showed a keen interest. He was interested in writing a chess-playing program. He wanted to know what expert aid Newell, Shaw, and Simon had enlisted in designing their chess-playing program. His expert aid would be Michael Tal, the former world's chess champion who lives in Riga.[15] The goal of his work will be to design a chess program to beat Michael Tal. He did not feel this was an impractical goal.

[15] A recent and fascinating article, M. Botvinnik, "Men and machines at the chessboard," *Soviet Rev.*, vol. 2, pp. 55–59; March, 1961, written by the world's chess champion, who is also an engineer, cites Tal as saying that it is impossible to build an intelligent chess machine. Botvinnik himself, I understand, is helping to program a chess machine

Arin's Speculation on Future Work

Arin and I discussed some of the more speculative ideas he has been considering. He proposes to model the human organism as an abstract machine and to search for the not-so-obvious properties of such a machine. Stated quite generally, he sees the human organism as a closed system, with input, output, and processes for self-organization, self-regulation, and self-maintenance. He proposes to model each separate subsystem of the body's machinery according to its particular function, focusing his analysis on the control of this machinery by a central computing device—his model of the central nervous system. Certain assumptions will be made about the nature of this central computing device and the machinery it controls—assumptions, for example, about its signal-transmission rate, rate of failure, rate of self-repair, and other properties of its self-regulation and self-maintenance machinery.

Arin posed the following question: Given a theory of the human machine constructed from a set of reasonable hypotheses about the body's machinery, one can deduce as a property of this machine that it is mortal? Or can it be proved that the machine might possibly live on indefinitely under certain conditions? Regardless of the chances of answering this question or others like it, from this approach Arin believes that the research will have great theoretical, and perhaps practical, interest. Of course, he wants to study not only the mortality properties of the system, but also other theoretical properties of self-regulating, self-maintaining closed systems.

Comments on Computers

Arin and I briefly discussed the present state of Soviet computers. He said that they were having difficulty perfecting the process for making reliable transistors, but that the problem was gradually being solved and they would soon have fully transistorized machines. He admitted that computer time was scarce, and, as I have already mentioned, he was somewhat worried about the future of his own research.

Evaluation and Conclusion

I shall try to summarize briefly my impressions of Dr. Arin. His understanding of the problems of building intelligent machines is genuine and deep. He has discovered the appropriate constructs and terminology for discussing problem solving by machines. His algebra machine cannot be classed now as an achievement of great lasting significance. But, on the other hand, it is neither trivial nor unimportant. Reflect on the fact that it was done in 1955 and 1956, roughly the time of the first successful heuristic problem-solving program written in this country.

There is the possibility that if Arin had had substantial computing power available to him during the past few years (instead of the use of only his very

small M-3), he and his students might have independently discovered the Newell-Shaw-Simon formulation of heuristic problem-solving programs for computers. But that is a big "if." The essential point is that Arin did not have this computing power available to him. His research has no computing priority. He does his research in his spare time, as it were. At the present time Arin has an administrative role which will further restrict his research activities.

VIII. Some Observations on the State of Computer Technology in the Soviet Union

I concur with the opinion of most U. S. computer scientists who have visited Russia that at present the U. S. has a definite lead over the Soviet Union in the design and production of computing machines, but that there is no gap in fundamental ideas, with the possible exception of the production of reliable transistors. With the importance of computers to modern science and technology, there is no doubt that fairly soon the Soviet Union will be producing as many computers as we do. To what extent they will utilize these computers effectively, and in what new ways, I have no immediate answer, but my trip did provide me with a few indications, as follows:

1) There appears to be a substantial body of high-caliber scientists and mathematicians, particularly in Moscow and Kiev, and perhaps in a few other places, who are informed, interested, and active in such research areas as cybernetic theory, brain models, learning machines, computer control of information, languages, and computer utilization in economic planning. If the reader wants to convince himself of this observation, I suggest that he scan the table of contents of the impressive five-volume series, "Problems in Cybernetics," edited by Lyapunov.[6] It is also not insignificant that the active motive force behind this work was, and still is, A. A. Lyapunov, one of the foremost mathematicians in the world today.

2) It appears that research in the computer sciences is directed centrally from Moscow. This unstartling observation is meant to imply that there seems to be little place for independent inquiry motivated by curiosity in this new and relatively unstructured research area. But where will the new ideas for the next generation of research come from? From the West? In that case, there will always be a lag between their research and ours.

Once the decision is made in Moscow to push a particular kind of research, resources can be concentrated. The mechanical translation effort is a case in point. Has the decision been taken to make a comparable effort in the area of artificial intelligence? This is not yet clear. There is some evidence that initial steps have been taken, but as yet the evidence from results is minimal.

3) Students are being trained on a broad scale in Moscow, Kiev, Leningrad, Riga—yes, even in Yerevan. This is an investment of great significance. Dr. Yershov himself was a student of Lyapunov in Moscow just a few years ago. The Soviet pool of competent scientists oriented toward cybernetics and advanced application of computers is growing. The full exercise of this potential will await a time when computing machine resources are less scarce and can be diverted with some priority to the research.

Epilogue

This report was written in late 1960. In a research area as dynamic as the computer sciences, it is inevitable that such a report is soon a bit out of date. I should like to update the report by mentioning a few Soviet developments which have recently come to my attention.

1) Volume V of "Problems in Cybernetics"[6] has been published. It includes a new section, "Problems in mathematical economics." In the Preface Lyapunov, the Editor, writes that mathematical economics " . . . is at present becoming very important," and cites a number of references to publications in linear programming (including translations of important American work in the field).

2) An article in the English-language *Moscow News* of August 12, 1961, entitled "The mechanical mind and creative endeavor," cites some Soviet research in artificial intelligence. Among the projects mentioned is that of Professor Amosov (Institute of Higher Nervous Activity, Kiev) on medical diagnosis by computer. A music composition program by R. Zaripov of the University in Rostov-on-Don is also described.

A complete translation of the research report on music composition with the URAL computer will be found in *Automation Express*, November, 1960. The program itself is not very interesting. It is significant, however, that in a provincial "small computer" city, research of this type is being given some priority for scarce computer time.

The same thought recurs with regard to reports of recent Russian attempts to program a chess-playing machine. During my trip, I discussed chess-playing programs with a number of Soviet scientists, and no one, with the exception of Dr. Arin, thought that these programs amounted to anything more than a frivolous waste of valuable resources. Yet a group, reported to contain Botvinnik, chess champion and engineer, is now attempting to write such a program, and computer time (on a BESM II, it is reported) is being allocated for this effort.

3) Though these few swallows do not yet make a summer, the implication is that computer time is becoming less scarce and that research on artificial intelligence and other advanced applications of computers is beginning to achieve a priority which it did not seem to have just one year ago.

Appendix I

Itinerary

Appendix II

Organization of the IFAC Conference

A. Theory

1) Theory of continuous linear systems
2) Theory of continuous nonlinear systems
3) Theory of discrete systems
4) Stochastic problems
5) Theory of optimum systems
6) Theory of self-adaptive systems
7) Theory of structures and methods of signal design
8) Special problems of mathematics
9) Simulation and experimental methods

B. Components

1) Electric and magnetic elements of control systems
2) Electric computing and analog devices, programming elements and controlling machines
3) Transducers, elements and systems of automatic and remote process control
4) Pneumatic automatic and computing devices
5) Devices and systems of automatic control

C. Industrial Applications

1) Automation in machinebuilding
2) Automation of power systems
3) Automation in chemical and oil-refining industries
4) Automatic electric drives and electric machines
5) Automation of metallurgical processes
6) Nonclassified problems

D. General Problems

Appendix III

Technical Excursions: Institutions Visited by Delegates of the 1st International IFAC Congress

Institution	*Excursion Program*
Moscow, June 27–July 7	
Institute of Automation and Telemechanics, USSR Academy of Sciences	Automatic control systems (adaptive, etc.). Telemechanical control systems (for oil fields, coal mines and power systems). Electric, pneumatic and hydraulic devices for automatic control systems. Analog computers.
Computation Center, U.S.S.R. Academy of Sciences	The "URAL" and "BESM II" electronic computers for mathematical problems.
Institute of Mining, U.S.S.R. Academy of Sciences	Automatic and remote control systems for the coal-mining and ore-mining industries.
Moscow State University	General principles of instruction and teaching methods. Work in progress at the chairs of theoretical mechanics and computer techniques.
Moscow Power Institute	General principles of instruction and teaching methods. Work in progress at the general science and specialized chairs.
First State Bearing Factory	Fully automatic production lines for the manufacture of bearings.
"Krasny Proletariy" Machine-Tool Factory	Integrated automatic gear manufacture line. Lathe assembly line.
Moscow Small-Power Car Factory	Automatic lines for machining engine parts.
Second State Watch and Clock Factory	Watch assembly line.
Power Station 20 of the Moscow Power System	Automatic control of boilers.
Load Dispatch Center of the Moscow Power System	Automatic and telemetering control of power systems.
Control Center of the Moskova Canal (Yakhroma, Moscow region)	Telemechanical control of pumping stations.
Leningrad, July 4–July 7	
Institute of Electromechanics, U.S.S.R. Academy of Sciences	Electro-dynamic analog for a power system, automatic control of telescopes, frequency converters.
Leningrad Institute of Electrical Engineers	General principles of instruction and teaching methods. Research program of general science and specialized chairs.
State Observatory, U.S.S.R. Academy of Sciences (Pulkovo)	Tour of the Observatory, astronomic instruments and devices used (a device for timing stars crossing the Pulkovo meridian, photoelectric coordinate reader). Laboratory of radioastronomy (radio-telescope).
"Vibrator" Plant	Manufacture of electrical measuring instruments: high-grade precision laboratory instruments, multi-channel recording oscillograph, photo-call instruments, manufacturing technology of miniature parts.
Sverdlov Machine-Tool Factory	Manufacture of program-controlled jig borers and semi-automatic machine-tools (copying machine-tools, etc.).
Distillery	Automated shop (washing, bottling, corking).
Kiev, July 4–July 7	
Computation Center, Ukrainian Academy of Sciences	The KIEV, URAL, and SESM electronic computers for mathematical problems.
The Paton Electric Welding Institute, Ukrainian Academy of Sciences	New welding techniques and equipment.
Institute of Automation, Ukrainian Academy of Sciences	Automatic control systems for iron-and-steel and chemical works, power systems, coal and ore mining, gas fields and engineering factories.
"Tochelectropribor" Plant	Process control in plastic parts shops, and assembly line for high-grade electric measuring instruments.

Acknowledgment

The author wishes to thank Paul Armer and Willis Ware of the Computer Sciences Department, the RAND Corporation, for their assistance in planning the trip, evaluating the material, and preparing the report.

Correspondence

Rapid Technique of Manual or Machine Binary-to-Decimal Integer Conversion Using Decimal Radix Arithmetic*

Often one has the problem of converting large binary integers to their decimal equivalent without the aid of a machine or conversion tables. Then one usually attacks the problem by expanding the series; adding in the binary or octal coefficients, multiplying by two or eight, respectively, starting with the most significant and proceeding to the least significant digit. Utilizing this series expansion another technique can be developed which is simpler and yields a result in relatively short order. (The author notes that he is not the originator, but to his knowledge this technique has not been previously published.) The technique seems novel and worth the attention of PGEC readers. The following steps describe the process:

1) Organize the binary digits in groups of three, inserting one or two zeros at most significant end if required.
2) Convert each three bit group to its octal equivalent.
3) Select most significant octal digit and double, using radix ten arithmetic.
4) Shift result of previous step right one position and subtract from operand.
5) Select most and next-to-most significant digits of newly formed operand, double using radix ten arithmetic, and repeat step four.
6) The final subtraction occurs when the shifted duplation registers its Least Significant Digit (LSD) under the LSD of the operand. The number in the desired radix is represented by the final operand, *i.e.*, the result of the final subtraction.

Summary: For each ith iteration, select the i most significant digits, double, shift right one position, and subtract from operand to form new operand. When LSDs of subtrahend and operand are in same position, the final operand is the decimal equivalent of the original octal number.

Analysis:

$$N_p \equiv \sum_0^n a_i \cdot p^i = a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + \cdots + a_0 \cdot p^0$$

$$N_q \equiv \sum_0^m b_i \cdot q^i = b_m \cdot q^m + b_{m-1} \cdot q^{m-1} + \cdots + b_0 \cdot q^0$$

where N_p is equivalent to N_q.

A statement of one iteration of the recursive process describing this proposed solution can be expressed.

$$N_i = (r_i \cdot q + a_{i-1} - \delta \cdot r_i) \cdot p^{i-1} + \sum_0^{i-2} a_i \cdot p^i,$$

where $r_i = a_n$, if $N_i = N_p$, also $\delta = q - p$.

$$N_i = (r_i \cdot q + a_{i-1} - r_i \cdot q + r_i \cdot p) \cdot p^{i-1} + \sum_0^{i-2} a_i \cdot p^i$$

$$N_i = (r_i \cdot p + a_{i-1}) \cdot p^{i-1} + \sum_0^{i-2} a_i \cdot p^i.$$

This final statement expresses one iteration of the well-known technique utilizing expansion properties of the known series. More rigid proofs are left to the reader.

Example:

Binary number	111,110,101,000,011,101,001.	
Octal number	7650351.	$7 \cdot 2 = 14$
1st iteration	-14	
1st iteration	6250351.	$62 \cdot 2 = 124$
2nd iteration	-124	
2nd iteration	5010351.	$501 \cdot 2 = 1002$
3rd iteration	-1002	
3rd iteration	4008351.	$4008 \cdot 2 = 8016$
4th iteration	$-\ 8016$	
4th iteration	3206751.	$32067 \cdot 2 = 64134$
5th iteration	$-\ 64134$	
5th iteration	2565411.	$256541 \cdot 2 = 513082$
6th iteration	$-\ 513082$	
6th iteration	2052329.	end, desired decimal number.

Conclusion: The technique described is simpler than the recursive process of adding in octal coefficients and multiplying by eight, because multiplication by eight is more complex than by two. The process of subtraction is comparable to addition except that numbers with more digits are subtracted.

One may note that this method is feasible for conversion of other radices. However, the factor, delta ($\delta = q - p$), may be altered.

JOHN E. CROY
International Electric Corp.
Paramus, N. J.

* Received by the PGEC, August 28, 1961.

A Note on Linear Separation*

McNaughton[1] has pointed out the need for an algorithm to determine whether two sets of points in an n-dimensional Euclidean space are linearly separated, *i.e.*, may be perfectly separated by a hyperplane in the n-dimensional space. He suggests an algorithm for points with binary coordinates which involves the solution of a set of linear inequalities. This note describes another computational algorithm, valid for points whose coordinates may take on values from either a continuous or discrete set, which involves the determination of the existence of a non-negative solution to a set of linear equalties.

The algorithm to be described depends on the following theorem and its corollary.

Theorem: Two sets of points are linearly separated if and only if their convex hulls are non-intersecting.[2]

Proof: The "if" part is a known result of linear algebra.[3] To prove the "only if," assume that the convex hulls of the linearly separated sets S_1 and S_2 intersect. There then exists at least one point p which is a convex combination of a set of points from S_1 and also of a set of points from S_2. It can be shown[4] that if each point of a set of points satisfies a linear inequality, then all convex combinations of these points also obey the inequality. Since there exists a hyperplane separating S_1 and S_2, this determines an inequality satisfied by points of S_1, but not by points of S_2. Then, since p is a convex combination of a set of points from S_1, p must satisfy this inequality. Since p is a convex combination of points from S_2, p must fail to satisfy this inequality. This is contradictory, and therefore S_1 and S_2 are not linearly separated.

Corollary: If there exists at least one point which is a convex combination of points from S_1 and which is also a convex combination of points from S_2, then S_1 and S_2 are not linearly separated.

An algebraic test for linear separation can be obtained from this corollary which leads to a set of $n+2$ linear equations in m unknowns, where n is the dimensionality of the space, and m is the total number of points in S_1 and S_2. If these equations have a non-negative solution, then the sets are not linearly separated. This set of linear equations is developed as follows.

Let p be a point contained in the convex hulls of both S_1 and S_2 (if such a point exists). Then p must be a convex combination of points from S_1 and also of points from S_2. Let there be m_1 points in S_1 and m_2 points in S_2. Let c_{jk} represent the kth coordinate of the jth point, where we distinguish between the sets by specifying that the range of j over S_1 is from 1 to m_1, and over S_2 is from m_1+1 to $m_1+m_2=m$. Then there is associated with each point a non-negative

* Received by the PGEC, July 5, 1961; revised manuscript received, August 14, 1961.

[1] R. McNaughton, "Unate truth functions," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 1–6; March, 1961.

[2] The convex hull of a set of points S is the smallest convex set containing S. It is the set of all convex combinations of sets of points from S.[4]

[3] A. E. Taylor, "Introduction to Functional Analysis," John Wiley and Sons, Inc., New York, N. Y.; 1958.

[4] S. I. Gass, "Linear Programming," McGraw-Hill Book Co., Inc., New York, N. Y.; 1958.

weight w_j such that

$$w_j \geq 0, \qquad 1 \leq j \leq m,$$

$$\sum_{j=1}^{m_1} w_j = 1$$

$$\sum_{j=m_1+1}^{m} w_j = 1,$$

and

$$p_k = \sum_{j=1}^{m_1} w_j c_{jk} = \sum_{j=m_1+1}^{m} w_j c_{jk}, \qquad 1 \leq k \leq n,$$

where p_k is the kth coordinate of p. This may be regarded as a set of $n+2$ equations in m unknowns (the unknowns being the weights w_j). If this set of linear equalities has a solution subject to the non-negativity constraint, then at least one such point p exists, indicating that the convex hulls overlap and consequently that the sets S_1 and S_2 are not linearly separated.

It has been pointed out by F. W. Sinden that this algorithm is the dual (in a linear programming sense) to McNaughton's algorithm, and that both may be handled with equal facility by the Simplex method of linear programming.[5] This method will give as a byproduct the separating hyperplane, if one exists.

Neither this algorithm nor that of McNaughton will give an optimum hyperplane if perfect separability is not possible. A rather involved algorithm for finding that hyperplane which correctly separates the greatest number of points in a continuous, discrete, or mixed space will be published soon.[6]

W. H. Highleyman
Bell Telephone Labs., Inc.
Murray Hill, N. J.

[5] D. Gale, "Theory of Linear Economic Models," McGraw-Hill Book Co., Inc., New York, N. Y.; 1960.
[6] W. H. Highleyman, "Linear Decision Functions, with Application to Pattern Recognition," Ph.D. thesis, E.E. Dept., Brooklyn Polytechnic Inst., Brooklyn, N. Y.; June, 1961.

Functional Notation for NOR and NAND Networks*

The present symbolic notation of NOR and NAND network functions involves, respectively, a dagger and a stroke symbol.[1] This note suggests an alternative symbolism employing functional notation. The functional notation proceeds from the viewpoint that the output of a NOR block with inputs A, B, and C is expressed as $N(A, B, C)$. Thus, in Fig. 1, $F_1{:}N(A, B, C)$ where the colon is to be read "is represented by." Similarly, $F_2{:}N[N(A, B, C)]=N^2(A, B, C)$ where the N^2 notation is used to denote the cascade of two NOR blocks where the only input to the second block is the output of the preceding block. Continuing with Fig. 1, it follows that $F_3{:}N[N^2(A, B, C)] = N^3(A, B, C) = N(A, B, C)$ since $N^2(\ )$ is an identity operation.

The functional notation readily allows one to obtain the NOR representation of the output (or any other) variable in the network. To see this consider Fig. 2 and the output F. One writes F as a function of the inputs which may be literals or the outputs of other NOR blocks which are expressed in functional form. Thus $F{:}N[C', N(\ )]$ which is in turn written $F{:}N[C', N(A, N[A, B'])]$. The expression for the output can usually be written immediately for networks more complicated than that of Fig. 2.

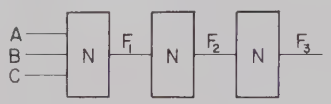


Fig. 1—Cascade of 3 NOR elements.

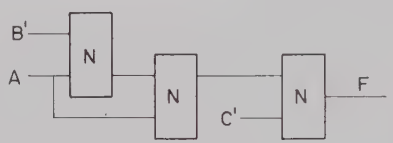


Fig. 2—Network for $F=(A+A'B)C$.

The functional notation permits a rapid method for converting a functional representation involving N to the form involving AND and OR operations. The conversion algorithm is derived from the fact that in Fig. 1, $F_1=(A+B+C)'=A'B'C'$ from which it follows that $ABC{:}N(A', B', C')$. Since a NOR block with identical inputs is a complementer, we have that $F_2=A+B+C$. Therefore, $A+B+C{:}N[N(A, B, C)] = N^2(A, B, C)$. Also, $A'{:}N(A)$. The AND and OR expressions are extendible to any number of variables.

Obtaining a functional representation of a Boolean function expressed in AND-OR form requires the systematic replacement of the AND and OR operators by their equivalent NOR representations. For example, the function $F=(A+A'B)C$ is converted by writing $F{:}N[C', (A+A'B)']$ but $(A+A'B)'{:}N[N^2(A, N[A, B'])]=N^3(A, N[A, B'])=N(A, N[A, B'])$ so $F{:}N[C', N(A, N[A, B'])]$. This last expression is the representation of the output of the network in Fig. 2. The conversion to functional form usually requires fewer steps than were used above after experience is acquired.

It will be observed that $F=(A+A'B)C$ can be simplified to $F=(A+B)C$ with the corresponding representation $N[C', N(A, B)]$. A more complete treatment[2] of the functional approach shows that by means of a table of equivalent representations this simpler representation can be obtained by substitution. It should also be noticed that once the functional expression has been obtained the resulting network can be laid out directly. The procedure is to start with the first N in the expression and work through the expression noting whether the variable in the argument of the function is an independent variable or the output of a preceding NOR block.

The conversion of a NOR representation to the equivalent AND-OR representation can be easily done in an operational manner. The technique proceeds on the basis that $N(A)=A'$ where the equals sign is to be interpreted in the sense that the output of a NOR block with an input A is A'. For $N(A, B, C)$ it follows that $N(A, B, C) = N(A)N(B)N(C)=A'B'C'$. Similarly, $N^2(A, B, C)=N[N(A, B, C)]=N[N(A)N(B)N(C)]=(A'B'C')'=A+B+C$. Equivalently, $N^2(A, B, C)=N^2(A)+N^2(B)+N^2(C)=A+B+C$ since $N^2(A)=N[N(A)]=N(A')=A$.

To illustrate the procedure, we will convert $F{:}N[C', N(A, N[A, B'])]$ to an equivalent AND-OR form. We begin with the first N and write according to the above rules $F=N(C')N^2[A, N(A, B')]=C(A+N^3(A, B'))$, but $N^3(A, B')=N^2[N(A, B')]=N(A, B')=A'B$ so we get $F=C(A+A'B)$. Similarly, $N[C', N(A, B)] = N(C')N^2(A, B)=C(A+B)$. The conversion procedure from NOR representation to the AND-OR form can be simplified with a little experience.

The colon was used to denote a NOR representation of some Boolean function. The equals sign was used in the conversion of the N expression to an AND-OR form to denote an operator equivalence such as $N(A)=A'$. The equals sign can actually be used in all cases since it is clear from the context whether the NOR representation or the operational equivalence is intended.

NAND networks can be treated in a manner similar to that discussed for NOR networks because of the duality between the NOR and NAND functions. Using $S(A, B, C)$ to denote the output of a NAND block with inputs A, B, and C it follows that $A'{:}S(A)$, $A+B+C{:}S(A', B', C')$, and $ABC{:}S^2(A, B, C)$. To obtain the NAND representation of $F=(A+A'B)C$, we replace the AND and OR operations by their S representations to get $F{:}S^2[C, S(A', S^2[A', B])]$. For $F=(A+B)C$ we get $F{:}S^2[C, S(A', B')]$. As in the case of the NOR networks, the first expression can be simplified to the second through a table of equivalent representations.[2]

The conversion of S representations to AND-OR form proceeds along the same lines as the NOR conversion. Thus, $S(A)=A'$, $S^2(A)=A$, $S^2(A, B, C)=S^2(A)S^2(B)S^2(C)=ABC$, and $S(A', B', C')=S(A')+S(B')+S(C')=A+B+C$. To illustrate the procedure, $S^2[C, S(A', B')] = S^2(C)S^3(A', B')=CS(A', B')=C(A+B)$.

The procedures discussed for the NOR and NAND networks are directly extendible to mixed networks containing NOR and NAND blocks.

Harold F. Klock
Case Institute of Technology
Cleveland, Ohio

* Received by the PGEC, December 5, 1960; revised manuscript received, August 7, 1961.
[1] N. Scott, "Analog and Digital Computer Technology," McGraw-Hill Book Co., New York, N. Y.; 1960.
[2] H. Klock, "Functional Synthesis and Analysis of NOR and NAND Networks," Engrg. Design Center, Case Inst. Tech., Cleveland, Ohio, Rept. No. EDC-1-61-3; January, 1961.

On the Algebraic Manipulation of Majority Logic*

One of the difficulties which arises in manipulating algebraic expressions involving majority logic is the fact that *ternary* operations are involved. In this note we would like to outline a method whereby this difficulty may be avoided. The method is based on a technique employed by Birkhoff and Kiss[1].

Let the majority function $M(X, Y, Z) = XY + XZ + YZ$ be written as $X \textcircled{Z} Y$. Then clearly $X \textcircled{Z} Y = Y \textcircled{Z} X$ and

$$X \textcircled{Z} Y = Z \textcircled{X} Y = X \textcircled{Y} Z. \quad (1)$$

Moreover, if in any Boolean identity, we replace $+$ by $\textcircled{Z}$, $\cdot$ by $\textcircled{Z}$, 1 by Z, and 0 by $\bar{Z}$, the resulting expression is equally valid under this new interpretation. Z may or may not appear as one of the variables in the original identity.

TABLE I

Boolean Identity	Majority Logic Identity
1) $X+\bar{X}=1$	1a) $X \textcircled{Z} \bar{X} = Z$
2) $X+X=X$	2a) $X \textcircled{Z} X = X$
3) $X+XY=X$	3a) $X \textcircled{Z} (X \textcircled{Z} Y) = X$
4) $X+\bar{X}Y=X+Y$	4a) $X \textcircled{Z} (\bar{X} \textcircled{Z} Y) = X \textcircled{Z} Y$
5) $\overline{X+Y}=\bar{X}\bar{Y}$	5a) $\overline{X \textcircled{Z} Y} = \bar{X} \textcircled{Z} \bar{Y}$
6) $X+0=X$	6a) $X \textcircled{Z} \bar{Z} = X$
7) $(W+X)+Y=W+(X+Y)$	7a) $(W \textcircled{Z} X) \textcircled{Z} Y = W \textcircled{Z} (X \textcircled{Z} Y)$
8) $W(X+Y)=WX+WY$	8a) $W \textcircled{Z} (X \textcircled{Z} Y) = (W \textcircled{Z} X) \textcircled{Z} (W \textcircled{Z} Y)$
9) $F(X, Z, \cdots)=X \cdot F(1, Z, \cdots) + \bar{X} \cdot F(0, Z, \cdots)$	9a) $F(X, Z, \cdots)=[X \textcircled{Z} F(Z, Z, \cdots)] \textcircled{Z} [\bar{X} \textcircled{Z} F(\bar{Z}, Z, \cdots)]$

Table I gives a list of some common Boolean identities and the corresponding majority logic expressions. It is interesting to note the similarity of this list with that given by Cohn and Lindaman.[2]

Another important relationship which does not result from the above substitution scheme is a *distributive* law

$$V \textcircled{Y} (W \textcircled{Z} X) = (V \textcircled{Y} W) \textcircled{Z} (V \textcircled{Y} X). \quad (2)$$

This method can also be extended to majority functions with any (odd) number of arguments by noting that

$$X_1 \textcircled{Y} X_2 \textcircled{Y} \cdots \textcircled{Y} X_n = M(X_1, Y, X_2, Y, \cdots, Y, X_n). \quad (3)$$

Sheldon B. Akers, Jr.
Information Processing
Electronics Laboratory
General Electric Co.
Syracuse, N. Y.

* Received by the PGEC, July 21, 1961. This work was supported under BuShips Contract NOBsr-81444.

[1] S. A. Kiss, "Transformations on Lattices and Structures of Logic," New York, N. Y., p. 184; 1947.

[2] M. Cohn and R. Lindaman, "Axiomatic majority-decision logic," IRE Trans. on Electronic Computers, vol. EC-10, pp. 17–21; March, 1961.

Conversion from Conventional to Negative-Base Number Representation*

Construction and application of negative-base number systems have been previously[1–4] discussed. For instance, the number represented by $-1{,}100.1$ in conventional binary notation would be represented by 110,100.1 (no sign required) in *negative* binary notation, because the first representation denotes $-(2^3+2^2+2^{-1}) = -12.5$ and the second denotes $(-2)^5+(-2)^4+(-2)^2+(-2)^{-1} = -12.5$.

To make a direct conversion from conventional binary to negative binary representation, application is made of the relations

$$\left.\begin{aligned}(+2)^n &= (-2)^n \\ -(+2)^n &= (-2)^{n+1}+(-2)^n\end{aligned}\right\} \quad n \text{ even}$$

$$\left.\begin{aligned}(+2)^n &= (-2)^{n+1}+(-2)^n \\ -(+2)^n &= (-2)^n\end{aligned}\right\} \quad n \text{ odd.}$$

The following procedures are thus indicated.

Positive numbers. Write down the conventional binary number. Underneath write a number consisting of "ones" in each even position which is one position to the left of a "one" in the original number, and "zeros" in all other positions. Add these two numbers in accordance with the rules[1] for "negative-binary" addition of digits:

$$0+0=0$$
$$0+1=1+0=1$$
$$1+1=110.$$

Example: If the conventional-binary number is given as 1,011.01, write

1,011.01	conventional-binary
+10,100.00	
Total 11,111.01	negative-binary.

* Received by the PGEC, June 20, 1961.

[1] L. B. Wadel, "Negative base number systems," IRE Trans. on Electronic Computers, vol. EC-6, p. 123; June, 1957.

[2] Z. Pawlak and A. Wakulicz, "Use of expansions with a negative basis in the arithmometer of a digital computer," *Bull. acad. polonaise sci.*, cl. 3, vol. 5, no. 3, pp. 233–236; March, 1957.

[3] W. Balasiński and S. Mrówka, "On algorithms of arithmetical operations," *Bull. acad. polonaise sci.*, cl. 3, vol. 5, no. 8, pp. 803–804; August, 1957.

[4] Z. Pawlak, "An electronic digital computer based on the "−2" system," *Bull. acad. polonaise sci.*, Série des sciences techniques, vol. 7, no. 12, pp. 713–721; December, 1959.

Negative numbers. Write down the absolute value of the conventional binary number. Underneath write a number consisting of "ones" in each odd position which is one position to the left of a "one" in the original number, and "zeros" in all other positions. Add these two numbers in accordance with the rules for "negative-binary" addition.

Example: If the conventional-binary number is given as $-1{,}101$, write

1,101	conventional-binary (absolute value)
+1,010	
Total 110,111	negative-binary.

Again note that no polarity sign is used in negative-base representations.

Louis B. Wadel
Vought Electronics
Arlington, Tex.

A New Method of Examining the Stability of Linear Systems Using a Repetitive Differential Analyzer*

Summary

The paper describes a method of examining the stability of linear systems which is based on a conformal transformation of the system equation and the application of Cauchy's principle of the argument. The method requires only the use of operational amplifiers of a repetitive analyzer and it has a number of important advantages over other methods which use analog-computing equipment. Using a very simple transformation, it is possible with this method to examine the stability of linear systems as well as to determine the damping constant. The results can be viewed directly on the screen of a cathode-ray oscilloscope.

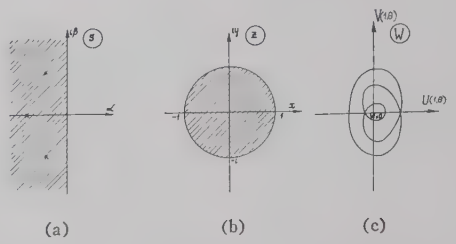
Introduction

In addition to the well-known numerical and graphical methods developed by Mitrović [1], Nyquist [2], Mikhailov [3] and others, several other methods of examining the stability of linear feedback systems by using repetitive analyzer and special-purpose analog computers are also available. In a previous paper [4] the author described a method based on the use of a repetitive differential analyzer and presented some other methods using special analog computers for the same purposes.

The purpose of this paper is to improve the method described in Petrić [4] and to simplify the use of the repetitive differential analyzer. Although the method described in this paper is based on principles which are

* Received by the PGEC, April 20, 1961.

similar to those described previously [4], it does not require the use of ganged linear potentiometers. Moreover, the answer to the question as to whether or not a linear system is stable can be obtained directly on a cathode-ray oscilloscope by viewing the results of the corresponding unit circle from the z plane to the W plane. In this regard, a suitable transformation was used which makes it possible to reduce the examination of the number of zeros of the corresponding algebraic polynomial lying in the left half-plane of the s plane to the examination of the number of zeros of the transformed polynomial lying in the unit circle. In this way the use of the repetitive differential analyzer has been greatly simplified.



(a) (b) (c)

Fig. 1—(a) s plane. (b) z plane. (c) W plane.

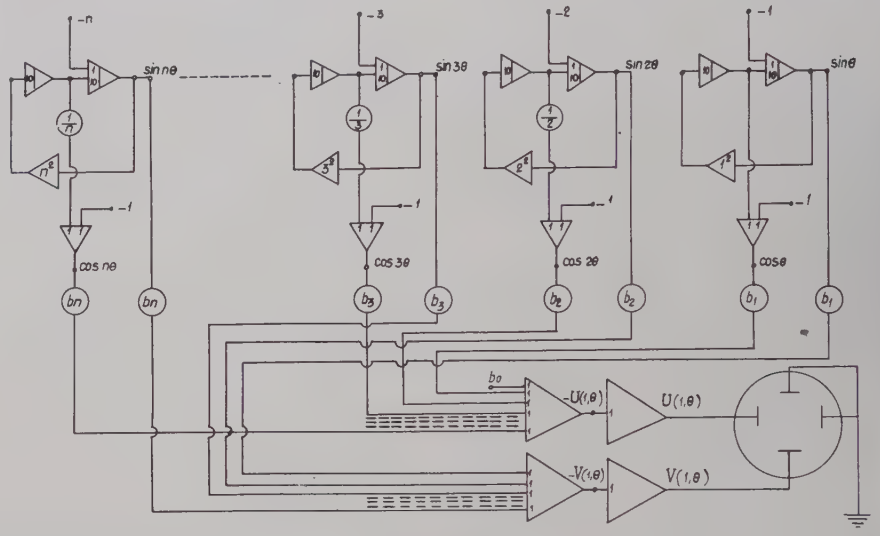


Fig. 2—General analog electrical model for conformal mapping of the unit circle by polynomial (4).

Basis of the Method

In the most general case where the characteristic polynomial of the linear system which is to be examined for stability is given in the form

$$W_0(s) = \sum_{\nu=0}^{n} a_\nu s^\nu \tag{1}$$

it is necessary to find out whether the condition

$$\mathrm{Re}\,(s_\nu) < 0 \tag{2}$$

is satisfied for all zeros of the polynomial (1). If the condition (2) is fulfilled, the system characterized by polynomial (1) is said to be stable. Otherwise the system is unstable or on the margin of stability.

Therefore, the answer to the question as to whether a given system is stable can be obtained either by knowing the values of zeros of polynomial (1), or by establishing through a suitable procedure how many zeros of polynomial (1) lie in the left half-plane of the s plane. Procedures for determining the zeros of polynomial (1) by the use of a repetitive differential analyzer are described in the literature [4], [5].

Employing the transformation

$$s = \frac{z+1}{z-1}, \tag{3}$$

the examination of stability of linear systems by examining the number of zeros of polynomial (1) lying in the left half-plane of the s plane [Fig. 1(a)] can be reduced to the examination of the number of zeros of the transformed polynomial

$$W(z) = (z-1)^n W_0\left(\frac{z+1}{z-1}\right) = \sum_{\nu=0}^{n} C_\nu z^\nu \tag{4}$$

lying in the unit circle $z=1$ [Fig. 1(b)].

In examining the stability of linear systems using the described method there are no difficulties when a differential analyzer is applied because the needed analog electrical model is very simple. If the variable z now traverses the unit circle, then, by Cauchy's principle of the argument, the path traced by W in the W plane will encircle the origin a number of times equal to the number of zeros inside the unit circle.

Coefficients of polynomial (4) can be determined by arranging the polynomial (1) according to increasing or decreasing powers of z after introducing transformation (3) and by using expression

$$C_{n-\kappa} = (-1)^\kappa \sum_{\nu=0}^{n} \left[\binom{n-\nu}{\kappa} - \binom{\nu}{1}\binom{n-\nu}{\kappa-1} + \binom{\nu}{2}\binom{n-\nu}{\kappa-2} - \binom{\nu}{3}\binom{n-\nu}{\kappa-3} + \cdots + (-1)^{\kappa-1}\binom{\nu}{\kappa-1}\binom{n-\nu}{1} + (-1)^\kappa\binom{\nu}{\kappa}\right] a_\nu,$$

$$\kappa = 0, 1, 2, \cdots, n, \tag{5}$$

which can easily be proved by Newton's binomial formula. Using expression (5) it is possible to determine that all coefficients of polynomial (1) are known, while taking into account that

$$\binom{\nu}{\kappa} = \frac{\nu!}{\kappa!(\nu-\kappa)!}. \tag{6}$$

Using substitution

$$z = \rho e^{i\theta}, \tag{7}$$

polynomial (4) can be written in the form

$$W(z) = U(\rho, \theta) + iV(\rho, \theta), \tag{8}$$

where

$$U(\rho, \theta) = \sum_{\nu=0}^{n} C_\nu \rho^\nu \cos \nu\theta \tag{9}$$

and

$$V(\rho, \theta) = \sum_{\nu=0}^{n} C_\nu \rho^\nu \sin \nu\theta. \tag{10}$$

For conformal mapping of the unit circle by polynomial (4), [Fig. 1(b)] in the W plane [Fig. 1(c)], expressions (9) and (10) are used in the form

$$U(1, \theta) = C_0 + C_1 \cos\theta + C_2 \cos 2\theta + \cdots + C_n \cos n\theta, \tag{11}$$

$$V(1, \theta) = C_1 \sin\theta + C_2 \sin 2\theta + \cdots + C_n \sin n\theta. \tag{12}$$

Analog electrical models for expressions (11) and (12) become very simple (Fig. 2) if cos $\nu\theta$ and sin $\nu\theta$ ($\nu = 1, 2, \cdots, n$) are represented by particular integrals of the differential equation

$$y'' + \nu^2 y = 0, \qquad (\nu = 1, 2, \cdots, n), \tag{13}$$

with initial conditions

$$y(0) = 0, \qquad y'(0) = \nu.$$

By viewing directly the results of conformal mapping of the unit circle by polynomial (4) on the screen of a cathode-ray oscilloscope, one can conclude whether the conditions for stability of a linear system described by polynomial (1) have been fulfilled.

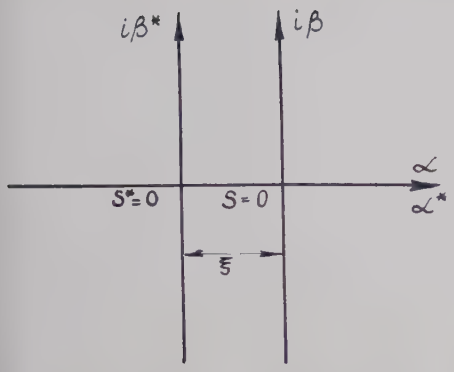


Fig. 3—s and s^* plane.

Fig. 4—The result of the conformal mapping of the unit circle using polynomial $W(z)$.

Determining the Damping Constant

If all zeros of polynomial (1) satisfy condition (2), the damping constant ξ of the system described by polynomial (1) is specified by the least absolute value of the real part of any zeros of polynomial (1). Therefore, the damping constant can easily be determined if the values of all zeros of polynomial (1) are known.

The method described in this paper, however, enables the damping constant to be determined without the need of determining the values of zeros of polynomial (1). To determine the damping constant ξ it is necessary to introduce substitution

$$s = s^* - \xi, \tag{14}$$

by which the axis of imaginaries is translated to the left by the amount ξ (Fig. 3). In this way it is possible to find the value ξ for which, after introducing transformation (14) into (1), the transformed polynomial will possess zeros which satisfy condition

$$\mathrm{Re}\,(s_\nu^*) \leq 0, \qquad \nu = 1, 2, \cdots, n. \tag{15}$$

Continuing the procedure of conformal mapping of the left half-plane of the s^* plane to the unit circle by using transformation.

$$s^* = \frac{z+1}{z-1},$$

one can determine the damping constant ξ within the limits of accuracy obtainable in solving standard problems on a repetitive differential analyzer. In practical application of differential analyzers for the determination of the damping constant ξ, the results obtained on the screen of a cathode-ray oscilloscope are of great importance. That value ξ for which one or simultaneously two of the closed loops are fading away on the screen is representing the damping constant of the examined linear system. It is only important that condition (15) is always satisfied, which is achieved by viewing on the cathode-ray oscilloscope the results of conformal mappping of the left half-plane of the s^* plane, *i.e.*, the corresponding unit circle.

Example

It is required to find out whether all zeros of polynomial

$$W_0(s) = s^3 + 4.1s^2 + 6.52s + 7.398$$

lie in the left half-plane of the s plane.

By using transformation (3) the left half-plane of the s plane is represented within the unit circle, and polynomial $W_0(s)$ is transformed into polynomial

$$W(z) = (z-1)^3 W_0\left(\frac{z+1}{z-1}\right)$$
$$= 19.018z^3 - 21.614z^2 + 14.574z - 3.978.$$

Using this polynomial for conformal mapping of the unit circle, the result shown in Fig. 4 is obtained, from which it is obvious that there are three closed loops around point $W=0$, *i.e.*, that all three zeros of the given polynomial lie in the left half-plane of the s plane.

The exact values of zeros of polynomial $W_0(s)$ are

$$s_1 = -2.7 \qquad s_{2,3} = -0.7 \pm 1,5i.$$

Conclusion

The method described in this paper requires the use of standard elements of only the linear part of a repetitive differential analyzer. Moreover, the method can be successfully used for solving the same problems on a digital computer.

The applied method has restrictions due to the narrow frequency band of operational amplifiers, so that it can only be used for solving problems in which $\nu \leq 7$.

Jovan Petrić
Automation Dept.
Inst. of Nuclear Sciences "Boris Kidač"
Belgrade, Yugoslavia

References

[1] G. J. Thaler and R. G. Brown, "Analysis and Design of Feedback Control Systems," McGraw-Hill Book Company, Inc., New York, N. Y., ch. 10; 1960.
[2] J. G. Truxal, "Automatic Feedback Control System Synthesis," Masgtz Moscow, USSR, pp. 142–149; 1959.
[3] N. N. Myasnikov, "Mikhailov's criteria," *Automatika i Telemekanika*, vol. 10, no. 4, pp. 267–273; 1949. In Russian.
[4] J. Petrić, "Trigonometric method for conformal mapping with algebraic polynomials by the use of repetitive differential analyser," in "Annales de l'Association Internationale pour le Calcul Analogique," Brussels, no. 1, pp. 11–17; 1961.
[5] P. Madić, I. Petrić and N. Parezanović, "The use of a repetitive differential analyzer for finding roots of polynomial equations," IRE Trans. on Electronic Computers, vol. EC-8, pp. 182–185; June, 1959.
[6] R. Tomović, "Calculateurs Analogique Répétitifs," Masson et Cie, Paris, France; 1958.

Correction to "A Modulo Two Adder for Three Inputs Using a Single Tunnel Diode"*

The author of the above correspondence item[1] wishes to call the following to the attention of the *Editor*.

In item 2, page 531, the last line should read, "the output current is high."

The last line of item 3, page 531, should read, "state C, and the output is low."

Karl S. Menger

* Received by the PGEC, November 13, 1961.
[1] IRE Trans. on Electronic Computers, vol. EC-10, pp. 530–531; September, 1961.

Minimization of Switching Circuits Subject to Reliability Conditions*

The minimization of switching circuits subject to reliability conditions is a generally difficult subject. The purpose of this note is to point out a relatively simple formulation for a restricted class of these problems.

The switching circuits considered here are of the two level "AND-to-OR" type. The failures considered are those which occur when a normally open switch may fail to close. The formulation of the problem is in terms of an integer linear program.

Conventional Minimization

Suppose a switching function has m minterms and n prime implicants. A vector $(a_{i1}, a_{i2}, \cdots, a_{in})$ can be associated with the ith minterm, where the a_{ij} are defined by:

$a_{ij}=1$ if the ith minterm implies the jth prime implicant
$=0$ otherwise.

Let there be n binary variables, $y_1, y_2, \cdots, y_n$, taking on the values:

$y_j=1$ if the jth prime implicant is selected
$=0$ otherwise.

The requirement that the ith minterm imply at least one prime implicant in the normal form is stated by the inequality

$$a_{i1}y_1 + a_{i2}y_2 + \cdots + a_{in}y_n \geq 1$$

and the requirement that the normal form be valid is given by the matrix relation

$$\begin{pmatrix} a_{11}a_{12}\cdots a_{1n} \\ a_{21}a_{22}\cdots a_{2n} \\ \cdot \\ \cdot \\ a_{m1}a_{mp}\cdots a_{mn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_n \end{pmatrix} \geq \begin{pmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ 1 \end{pmatrix}$$

* Received by the PGEC, November 15, 1960, revised manuscript received, August 7, 1961.

or, more compactly, $\boldsymbol{Ay} \geq \boldsymbol{1}$.[1] The matrix $\boldsymbol{A}=(a_{ij})$ will be referred to hereafter as a *constraint matrix*. It corresponds to the transpose of the conventional prime implicant chart.

The minimization requirement is expressed with the aid of an appropriate cost vector $(c_1, c_2, \cdots, c_n)$. For example, c_j may be set equal to the number of literals in the jth prime implicant. The prime implicant selection problem is seen to be of the form

$$\begin{aligned} &\text{minimize } \boldsymbol{cy} \\ &\text{subject to } \boldsymbol{Ay} \geq \boldsymbol{1} \\ &\text{and } \quad y_j = 0, 1 \quad (j = 1, 2, \cdots, n). \end{aligned}$$

This happens to be a standard linear program, with the exception that the variables are restricted to discrete values, rather than non-negative values.

Minimization with Reliability Conditions

Gill[2] has discussed the problem of obtaining a minimal normal form subject to the condition that every minterm of the function implies at least t terms of the normal form. In a circuit in which each term is realized by an AND gate, this condition assures that at least t such gates must fail before failure of the circuit is possible.[3]

It is easily verified that a solution to Gill's problem is found entirely in terms of prime implicants. Using the notation introduced above, it is necessary to

$$\begin{aligned} &\text{minimize } \boldsymbol{cy} \\ &\text{subject to } \boldsymbol{Ay} \geq \boldsymbol{t} \\ &\text{and } \quad y_j = 0, 1, 2, \cdots, t \\ &\qquad\qquad (j = 1, 2, \cdots, n), \end{aligned}$$

where the constraint matrix $\boldsymbol{A}$ and each of the vectors is defined as before. The value of y_j indicates the number of times the jth prime implicant is included in the solution.

A refinement in the formulation of the reliability problem is possible. Suppose that the switching function is to be realized as a contact network, and that each contact represents a normally open switch that may fail to close with probability ϵ, independently of each of the other contacts. Such switches might be called "semi-crummy," since they never fail to reopen properly, unlike the "crummy" relays of Moore and Shannon.[4]

A series path of c contacts will fail to transmit when one or more of the contacts fails to close. The (conditional) probability of this event is $1-(1-\epsilon)^c$. If m series paths of $c_1, c_2, \cdots, c_m$ contacts are all supposed to transmit for a given combination of values of the variables, the (conditional) probability of failure of the circuit for that combination is

$$\prod_{j=1}^{m} [1 - (1 - \epsilon)^{c_j}].$$

The logarithm of the probability of failure of the circuit is

$$\sum_{j=1}^{m} \log [1 - (1 - \epsilon)^{c_j}].$$

Suppose that a contact network is desired whose probability of failure for each combination of the input variables is less than some value $0<\eta\leq 1$. It is again easily verified that a minimal solution to this problem is found entirely in terms of prime implicants. The prime implicant selection problem can be formulated as follows:

Let $\boldsymbol{A}$ be a constraint matrix, as before, except that the elements of $\boldsymbol{A}$ are defined as follows:

$$a_{ij} = \frac{\log [1 - (1 - \epsilon)^{c_j}]}{\log \eta} \text{ if } i\text{th minterm implies } j\text{th prime implicant}$$

$$= 0 \text{ otherwise,}$$

where c_j is the number of literals in the jth prime implicant. Then it is necessary to

$$\begin{aligned} &\text{minimize } \boldsymbol{cy} \\ &\text{subject to } \boldsymbol{Ay} \geq \boldsymbol{1} \\ &\text{and } \quad y_j = 0, 1, 2, \cdots \quad (j = 1, 2, \cdots n). \end{aligned}$$

The value of y_j indicates the number of times the series path represented by the jth prime implicant is to be included in the contact network.

Note that

$$\lim_{\epsilon,\eta \to 0} \left\{ \frac{\log [1 - (1 - \epsilon)^{c_j}]}{\log \eta} \right\} = 1,$$

so that for the case of perfectly reliable components this problem reduces to the ordinary prime implicant selection problem.

Further refinements can be made. In particular, it is possible to assign differing *a priori* probabilities to the various input combinations. The network can then be minimized according to a specified *expectation* of failure.

Computational methods have been proposed by Gomory,[5] and by Land and Doig,[6] for solving integer linear programs. It is also usually possible to obtain a reasonably economical, although not demonstrably minimal, solution by an adaptation of conventional prime implicant selection or cut-and-try methods.

Example: Consider the 4-variable function with minterms 0, 1, 2, 4, 7, and 8 and "don't cares" 10, 11, 12, 13, 14, 15. Any minimal normal form for this function must be composed of the following prime implicants:

Prime Implicant	*Minterms Covered*
$x_2x_3x_4$	7
$\bar{x}_1\bar{x}_2\bar{x}_3$	0, 1
$\bar{x}_3\bar{x}_4$	0, 4, 8
$\bar{x}_2\bar{x}_4$	0, 2, 8
$x_1\bar{x}_4$	8

Suppose that a circuit is to be designed in which the probability of failure of any contact is $\epsilon=\frac{1}{2}$, and that the entire circuit is to have a probability of failure no greater than $\eta=1/100$. Application of the formulas derived above gives the problem:

$$\begin{aligned} \text{minimize } \quad & 3y_1 + 3y_2 + 2y_3 + 2y_4 + 2y_5 \\ \text{subject to } \quad & 0.029y_2 + 0.062y_3 + 0.062y_4 \geq 1 \\ & 0.029y_2 \geq 1 \\ & 0.062y_3 \geq 1 \\ & 0.062y_4 \geq 1 \\ & 0.029y_1 \geq 1 \\ & 0.062y_3 + 0.062y_4 + 0.062y_5 \geq 1 \\ \text{and } & y_j = 0, 1, 2, \cdots \quad (j = 1, 2, \cdots, 5). \end{aligned}$$

It can be seen by inspection that an optimal solution to the linear program is obtained by setting $y_1=y_2=35$ and $y_3=y_2=17$. Thus a contact network meeting the specified reliability conditions contains 35 repetitions of the series paths $x_2x_3x_4$ and $\bar{x}_1\bar{x}_2\bar{x}_3$ and 17 repetitions of $\bar{x}_3\bar{x}_4$ and $\bar{x}_2\bar{x}_4$.

E. L. Lawler
Sylvania Electric Products
Needham, Mass.

[1] Vector inequality is a component by component relation: $\boldsymbol{a}\geq\boldsymbol{b}$ if, and only if, $a_j\geq b_j$ for all j. The symbol $\boldsymbol{1}$ denotes an m-dimensional column vector, all of whose components are the scalar $\boldsymbol{1}$.

[2] A. Gill, "Minimization of contact networks subject to reliability specifications," IRE Trans. on Electronic Computers (Correspondence), vol. EC-9, pp. 122–132; March, 1960.

[3] This assumes that the only type of failure of an AND gate is the failure to produce a 1 as output when all its inputs are 1.

[4] E. F. Moore and C. E. Shannon, "Reliable circuits using less reliable relays," *J. Franklin Inst.*, vol. 262, pp. 191–208, September; pp. 281–297, October, 1956.

[5] R. E. Gomory, "An Algorithm for Integer Solutions to Linear Programs," Princeton-IBM Math Res. Project, Princeton, N. J., Tech. Rept. No. 1, 1958.

[6] A. W. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520; July, 1960.

Variable Time Delay by Padé Approximation*

Up to this time, work in variable time delay has been done by Padé approximation for fixed-time delays. For variable time delays, most authors have proposed magnetic drum or tape with movable heads to adjust this delay. This method suffered from a lack of flexibility in that the time delay could not be made proportional to a variable as represented by another voltage. In this note, Padé approximation was used to get the desired time delays. However, the ratios which involved the delay were not fixed by potentiometers, but were generated by servo dividers. This permitted the delay to be entered as a voltage. Because at a delay of zero an infinite response would be obtained, a finite delay was always entered into the system. The second-order system was wired up on a PACE 16-31R and was found to give a discernible step function response up to 10 seconds delay, but rise time suffered. It was also noted that there was an initial undershoot in the response. Sine waves were reproduced without distortion.

It is felt by the author that much better step function response could be obtained with the fourth-order circuit shown in Figs. 2 and 4. With time delay now available as a variable in problem solutions, further work may point the way to solve boundary value problems.

Subsequent to the original work on this note, circuits have been used which use fewer amplifiers and ganged pots; however, they require a servo-setting capacitor.

It is felt that these circuits discussed here would be of value to people who do not have special equipment. It should be noted that other people have also had oscillatory difficulties with responses to step functions.

The author wishes to thank R. Beaudoin for his help in this project.

David L. Zackon
McGill University
Montreal, Quebec, Can.

Bibliography

[1] C. L. Johnson, "Analog Computer Techniques," McGraw-Hill Book Co., Inc., Toronto, Ontario, Can., pp. 71–72, 131–132.
[2] W. J. Cunningham, "Time delay networks for an analog computer," IRE Trans. on Electronic Computers, vol. EC-3, pp. 16–18; December, 1954.
[3] A. S. Jackson, "Analog Computation," McGraw-Hill Book Co., Inc., New York, N. Y., pp. 255–259.

* Received by the PGEC, June 29, 1961.

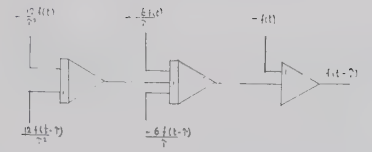


Fig. 1—Second-order Padé approximation circuit.

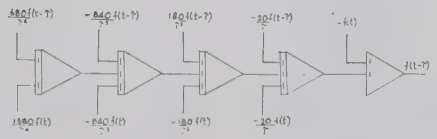


Fig. 2—Fourth-order Padé approximation circuit.

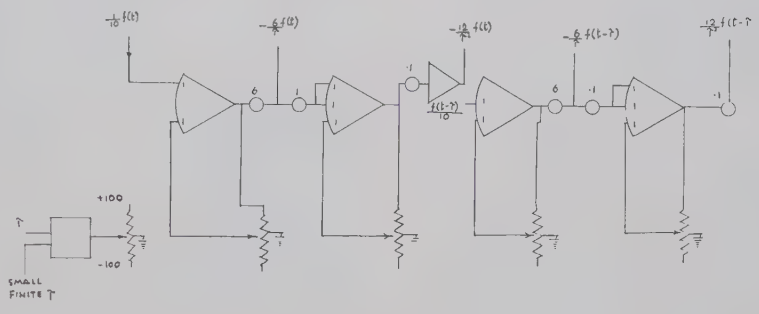


Fig. 3—Second-order division circuit for Padé approximation.

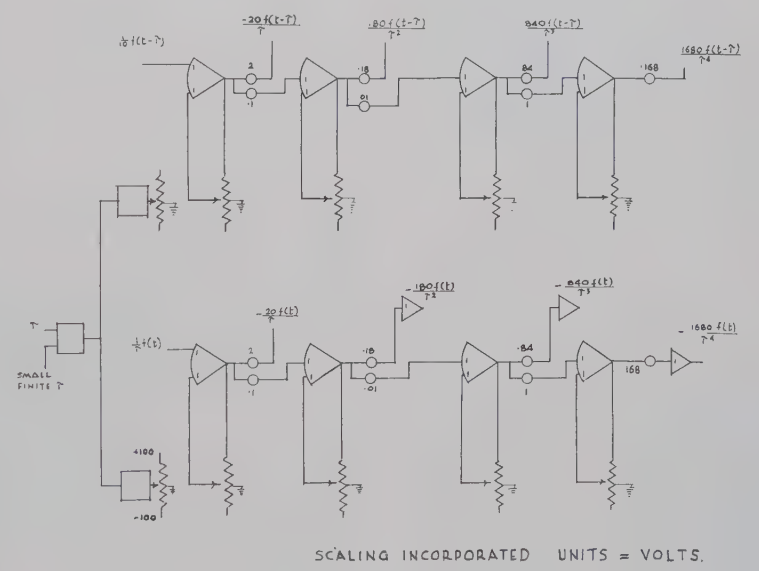
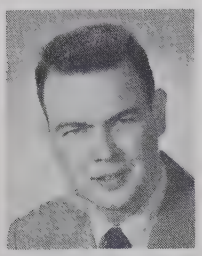


Fig. 4—Fourth-order division circuit for Padé approximation.

Contributors

Sheldon B. Akers, Jr., was born in Washington, D. C. on October 22, 1926. He received the B.S. degree in electrical engineering and the M.A. degree in mathematics at the University of Maryland, College Park, in 1948 and 1952, respectively.



S. B. Akers, Jr.

From 1948 to 1950 he was employed as an electronic scientist in the Central Radio Propagation Laboratory of the National Bureau of Standards, Washington, D. C. From 1950 to 1953 he worked as a Radio Engineer in the Electronics Division of the U. S. Coast Guard Headquarters, Washington, D. C. In 1953 he joined the Computer Section of the National Bureau of Standards where he was associated with the construction of the DYSEAC computer. In 1954 he transferred to the Avion Division of the ACF Industries in Alexandria, Va., where he headed a group primarily concerned with an Air Force contract for the mechanization of deductive reasoning. In the fall of 1956 he accepted his present position as a mathematician in General Electric's Electronics Laboratory, Syracuse, N. Y. Here his work has dealt mainly with logical design, theory of automata, and operations research.

Mr. Akers is a member of Pi Delta Epsilon, Omicron Delta Kappa, The Scientific Research Society of America, and The American Mathematical Society.

❖

Merwyn E. Arthur was born in Galena, Kan., on September 1, 1909. He attended Flat River Municipal Junior College, Flat River, Mo., and the University of Kansas, Lawrence.

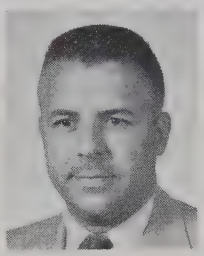
M. E. Arthur

In 1951, he joined IBM as a Design Engineer in Endicott, N. Y., where he worked with a group responsible for devising a system for the automatic inspection of cards being printed on a rotary press. He became Associate Engineer at the IBM Space Guidance Center, Owego, N. Y., in 1955. In this capacity he designed the control circuitry of the servo unit test set of the MA-2 System. Promoted in 1957 to his present position of Staff Engineer, he designed the logic for the automatic drum loader for the B-70 bombing and navigation system and was responsible for the design of a data controller, which transfers information from a digital computer to another type computer. He now is responsible for designing the logic of a variety of equipments. He has been active in originating nomograms, some of which have been published.

Mr. Arthur was co-winner of the American Society for Engineering Education, Graphic Science Division, Nomography Award for 1960.

❖

Carl L. Becker (A'55–M'60) was born in Cressona, Pa. on July 8, 1924. He received the B.S.E.E. degree from Brown University, Providence, R. I., and the M.S.E.E. degree from Carnegie Institute of Technology, Pittsburgh, Pa. in 1945 and 1947, respectively. He is currently working toward the Ph.D. degree in electrical engineering at the University of Arizona, Tucson.



C. L. Becker

From 1947 to 1950 he was an Assistant Professor of electrical engineering at Clemson College, Clemson, S. C. and from 1950 to 1952, he was a Teaching Assistant in electrical engineering while a graduate student at the Massachusetts Institute of Technology, Cambridge. In 1952, he became a staff member at M.I.T.'s Lincoln Laboratory, Lexington, Mass., where he was engaged in development work on Doppler radar. He became a member of the Research Department of Melpar, Inc., Cambridge, Mass., in 1954. Since 1955, he has been with the Radio Corporation of America, Tucson, Ariz., where he has been concerned with communication-system engineering problems. He is currently on leave of absence from RCA.

Mr. Becker is a member of Tau Beta Pi, Sigma Xi, and Sigma Pi Sigma.

❖

Don R. Boyle was born in Toledo, Ohio, on July 8, 1936. He received the B.S. degree in electrical engineering in 1958 from the University of Maryland, College Park.

D. R. Boyle

Since 1954, he has been employed at the National Bureau of Standards in Washington, D. C., working in digital technology. His interests include new devices, circuit design and simulation, and logical design of automatic data logging systems.

He is a member of Phi Eta Sigma, Phi Kappa Phi, Eta Kappa Nu, and Tau Beta Pi.

Robert C. Brigham was born in East Orange, N. J., on September 20, 1934. He received the B.S. and M.S. degrees in electrical engineering in 1957 from the Massachusetts Institute of Technology, Cambridge. He was a Fulbright Scholar from 1957 to 1958 at the University of New South Wales, Sydney, Australia, where he did research on the University's DEUCE computer.

R. C. Brigham

He is currently with the Martin Company, Orlando, Fla., and is engaged in several studies involving computer applications.

Mr. Brigham is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.

❖

Naphtali Burla was born in Tel Aviv on May 30, 1931. After service in the Israel Defence Forces he joined the Israel Post Telephone and Telegraph Service in 1951, working as a telephone exchange maintenance technician. He received the B.Sc. degree in electrical engineering from The Technion, Israel Institute of Technology, Haifa, in 1956.

N. Burla

After two more years with the Telephone Service, he returned to the Graduate School of The Technion, as Research Assistant, in 1958. He received his Master's degree from The Technion, for a thesis, "Some Problems in The Design of High Speed Binary Arithmetic Units" in June, 1960.

❖

Charles Coleman was born in Washington, D. C., on December 20, 1934. He majored in mathematics and physics at the University of Virginia, Charlottesville.

C. Coleman

From 1955 to 1961 he worked at the National Bureau of Standards, Washington, D. C., on the application of computing machines to the solution of problems in optics, spectroscopy, atomic physics, and computer design. In 1958–1959, as a member of the Digital Systems Section of NBS, he wrote the programs

which were used for converting the logical design of the NBS PILOT multiple computer system into detailed wiring plans and maintenance manuals. In 1960 and 1961, he held the post of lecturer on the structure of computing machines at the summer institute for college teachers of mathematics, engineering and physical sciences, at the University of California, Los Angeles. In October 1961, he accepted a position with the International Business Machines Corporation Research Center, Yorktown Heights, N. Y.

Mr. Coleman is a member of the Optical Society of America, the ACM, and the American Physical Society.

❖

Charles S. Deering (A'55–M'56) was born in San Antonio, Tex., on March 4, 1931. He received the B.S. and M.A. degrees in physics from the University of Texas, Austin, in 1952 and 1955, respectively, and the M.S. degree in electrical engineering from Southern Methodist University, Dallas, Tex., in 1959.

C. S. Deering

Concurrent with graduate study, he worked on the development of acoustic filters at Defense Research Laboratory, Austin, Tex. From 1953 to 1959 he was employed by Convair, Fort Worth, Tex., where he was engaged in the development of analog and digital computing circuitry. In late 1959 he joined Chance Vought Corp., Arlington, Tex., where he is employed as an Engineering Specialist. His responsibilities there have consisted of the logical and system design of both incremental and full-word-length digital computers.

❖

Sidney N. Einhorn was born in Philadelphia, Pa., on July 13, 1928. He received the B.S. degree in electrical engineering from the Drexel Institute of Technology, Philadelphia, in 1950, and the M.S. degree in electrical engineering from the University of Pennsylvania, Philadelphia, in 1958.

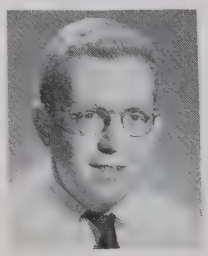
S. N. Einhorn

From 1953 to 1954 he was on the staff of the Moore School of Electrical Engineering at the University of Pennsylvania, where he conducted research on high-speed digital techniques. Since 1954 he has been associated with the Burroughs Corporation Research Center, Paoli, Pa., where he has been active in research on magnetic devices; he has two patent applications in this field. In the design field, Mr. Einhorn is responsible for the driver and addressing circuitry for a coincident-current memory; for the electrical design integrating conventional diode logic and RCTL, as well as the accumulator and associated circuits for a test vehicle; and for a decimal accumulating shift register which uses a multiple stable-state magnetic counter.

Mr. Einhorn is a member of RESA.

❖

Edward A. Feigenbaum was born in Weehawken, N. J., on January 20, 1936. He received the B.S. degree in electrical engineering in 1956 and the Ph.D. degree in industrial administration in 1960, both from Carnegie Institute of Technology, Pittsburgh, Pa. He spent the 1959–1960 academic year at the British National Physical Laboratory, Feddington, Middlesex, on a Fulbright fellowship.



E. A. Feigenbaum

He is Assistant Professor in the School of Business Administration, University of California, Berkeley, and Consultant to the Computer Sciences Department, the RAND Corporation, Santa Monica, Calif. His research has been in the areas of artificial intelligence, computer simulation of human learning processes, and list-processing computer languages. At Berkeley he holds research appointments in the Management Sciences Research Center and the Center for Human Learning.

Dr. Feigenbaum is a member of the Association for Computing Machinery and the AAAS.

❖

David N. Freeman was born in Boston, Mass., on September 29, 1933. He received the B.A. degree in mathematics in 1955, from Yale University, New Haven, Conn., and the M.S. degree in mathematics in 1958, from Cornell University, Ithaca, N. Y.

D. N. Freeman

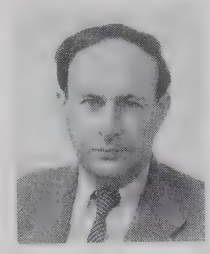
In September, 1958, he joined International Business Machines Corporation as a mathematician at the company's Development Laboratory in Endicott, N. Y. After gaining experience as an IBM 704 programmer in the Scientific Computation Laboratory, Endicott, he received an IBM fellowship for advanced study. He is now working towards a Ph.D. degree in operations research at Cornell University, Ithaca, N. Y.

Mr. Freeman is a member of Sigma Xi, Phi Kappa Phi, the American Mathematical Society, the Operations Research Society of America, and the Institute of Management Services.

❖

Ephraim H. Frei (M'58) was born in Vienna, Austria, on March 2, 1912. He received the Ph.D. degree from the University of Vienna, Austria, in 1936.

With the exception of the years 1950 through 1952, which were spent with the Computer Project of the Institute for Advanced Study, Princeton, N. J., he has been with the Department of Electronics, Weizmann Institute of Science, Rehovoth, Israel, since 1948. He was appointed Professor in 1960, and Head of Department in 1961. During the year 1959–1960, while on sabbatical leave, he was at the Stanford Research Institute, Menlo Park, Calif., where he worked on magnetic computer devices and measurements.



E. H. Frei

Dr. Frei is a member of the Israel Physical Society and the American Physical Society.

❖

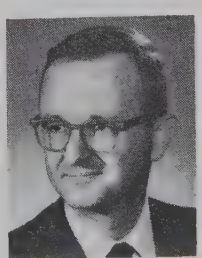
Sidney B. Geller was born in New York, N. Y., on May 6, 1920. He attended The George Washington University, the University of Florence, Italy, and the National Bureau of Standards Graduate School, Washington, D. C., where he studied mathematics and electronics engineering. He also studied television engineering with the Capitol Radio Engineering Institute, Washington, D. C.

S. B. Geller

From 1950 to 1954 he was a television engineer with the National Broadcasting Company and the American Broadcasting Company. Since 1954 he has been with the National Bureau of Standards, where he has worked as an electronics engineer in the design of miniaturized radar equipment, and transistorized control circuitry. From 1958 to the present time he has been engaged in the study of semiconductor devices.

❖

Jacob Goldberg (S'50–A'50–M'56) was born in San Francisco, Calif. on June 4, 1926. He received the B.S.E.E. degree in 1950, from the University of California, Berkeley, and the M.S.E.E. degree in 1954, from Stanford University, Stanford, Calif. In 1958, he was a Research Fellow at The Weizmann Institute, Rehovoth, Israel.



J. Goldberg

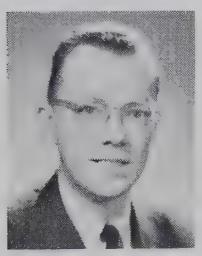
Since 1951, he has been a staff member of Stanford Research Institute, Menlo Park, Calif., where he is presently a Senior Research Engineer. His major interest has been

the design of digital computer systems. He was one of the logical designers of the ERMA computer and is currently engaged in the study of special file structures for data retrieval, and the design of reliable computer systems.

Mr. Goldberg is a member of the Scientific Research Society of America, and the Association for Computing Machinery.

❖

Bert F. Green, Jr., was born on November 5, 1927, in Honesdale, Pa. He received the B.A. degree from Yale University, New Haven, Conn., in 1949, and the M.A. and Ph.D. degrees in 1950 and 1951, respectively, from Princeton University, Princeton, N. J.



B. F. Green, Jr.

While at Princeton, he was a Psychometric Fellow at the Educational Testing Service. In 1951, he joined the Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass., where he has been leader of the Psychology Group since 1959. He is now on leave from M.I.T. to serve as consultant to the Computer Sciences Dept., RAND Corporation, Santa Monica, Calif. His present work with computers is concerned with improving man-machine communication. He has published papers on engineering psychology, visual perception, psychometrics, statistics, and the use of computers in psychology.

Dr. Green is a fellow of the American Psychological Association, and a member of the Psychometric Society and the Association for Computing Machinery.

❖

Lee E. Hargrave, Jr. (S'56–M'60), was born in Herrin, Ill., on October 31, 1934. He received the B.A. degree in mathematics in 1956 and the B.S. degree in electrical engineering in 1957, both from the University of Pennsylvania, Philadelphia, and the M.S. degree in electrical engineering from the University of Maryland, College Park, in 1960.

L. E. Hargrave

In 1956 he was Junior Engineer with the Electronics and Armament Systems Division of Lockheed Aircraft Corporation, Burbank, Calif., where he applied the digital computer to calculations of airborne radar propagation. From 1956 to 1957 he was Student Engineer with Remington Rand Univac in Philadelphia, where he was engaged in circuit design for the Univac Larc computer. As Electronic Engineer with the Department of Defense from 1957 to 1960, he participated in research and development of radio terminal equipment. From 1958 to 1960 he was Instructor in Transistor Circuitry with the University College of the University of Maryland. He is currently associated with Sanders Associates, Inc., of Nashua, N. H., as Senior Electronic Engineer in preliminary design.

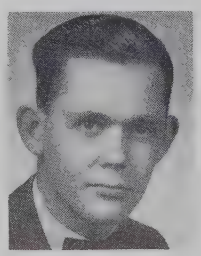
Mr. Hargrave is a member of Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, Sigma Tau, and Pi Mu Epsilon.

❖

Juris Hartmanis, for a photograph and biography, please see page 293 of the June, 1961, issue of these Transactions.

❖

Per Asbjørn Holst was born in Strand, Norway on August 8, 1932. He received the M.S. degree in electrical engineering in 1956 from the Norwegian Institute of Technology, Trondheim, Norway.

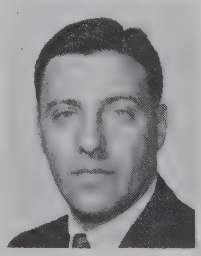


P. A. Holst

Since 1957 he has been employed by the Chr. Michelsens Institute, Dept. of Applied Physics, Bergen, Norway, as a research physicist, where he has been working on servomechanisms and process control, mainly in connection with industrial applications. In 1959 he was granted a scholarship from N.T.N.F., the Royal Norwegian Council of Technical and Industrial Research, for post-graduate studies in switching circuits and digital engineering at Massachusetts Institute of Technology, Cambridge, where he remained as a guest of the Research Laboratory of Electronics during the 1960–1961 academic year.

❖

Gerhard L. Hollander (S'43–M'47–SM'56) was born in Berlin, Germany, on February 27, 1922. He received the B.S.E.E. degree from the Illinois Institute of Technology, Chicago, in 1947; the M.S.E.E. degree from Washington University, St. Louis, Mo., in 1948; and the advanced professional E.E. degree from the Massachusetts Institute of Technology, Cambridge, in 1953.



G. L. Hollander

He heads Hollander Associates of Fullerton, Calif. which he organized to provide design and consulting services in computer and automatic control systems. He formerly headed system management and computer operations at the Philco Corporation, Philadelphia, Pa., and at the Hughs Aircraft Company, Fullerton, and has been a consultant to Headquarters, U. S. Air Force, on control system integration. He has guided the development of the Tape-DRUM, the C-1100 airborne computers, and the C-3000 control computer. His recent activities include gigahertz logic, the MPC-1200 multiple-processor computer, and major defense and communication systems. During World War II he was awarded a Navy commendation for developing a radar training device for the MK-57 fire-control system. He established the electrical engineering curriculum at St. Louis University, St. Louis, Mo., and designed the forerunner of present digital recorders at the M.I.T. Servomechanisms Laboratory.

Mr. Hollander is a member of AIEE, Sigma Xi, headed the AIEE Digital-Computer and Computer-Systems Subcommittees, and is presently Vice Chairman of the AIEE Computing Devices Committee, and a member of the National Joint Computer Committee. He was a U. S. delegate to the IFAC Congress in Moscow, and is on the Control Advisory Committee of the American Automatic Control Council. He is a registered Professional Engineer and editor of the book *Computers in Control.*

❖

Ralph C. Johnston (S'53–M'57) was born in Fremont, Neb. on May 22, 1933. He received the B.S. degree in electrical engineering from Iowa State College, Ames, in 1955. The following year he was a National Science Foundation Fellow at the Massachusetts Institute of Technology, Cambridge. From 1956 to 1958, he was connected with the M.I.T. Lincoln Laboratory, Lexington, as a staff associate and in 1958 received the S.M. degree in electrical engineering and the degree of Electrical Engineer from M.I.T.

R. C. Johnston

Presently a staff member at the Lincoln Laboratory, he has been active in the fields of HF transistor measurements, and circuits for UHF computers.

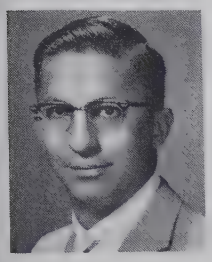
Mr. Johnston is a member of Tau Beta Pi and Eta Kappa Nu.

❖

Robert S. Ledley (M'58) was born on June 28, 1926, in New York, N. Y. He received the D.D.S. degree in 1948 from New York University College of Dentistry, and the M.A. degree in mathematical physics in 1949 from Columbia University, New York, N. Y.

He joined the National Bureau of Standards, Washington, D. C., in 1951, working in biophysics and later on the logical design and other aspects of digital computers. In 1954 he became a staff member of the Opera-

tions Research Office of The Johns Hopkins University, where he worked on computer simulations and the application of logic to operations research problems. In 1956 he became an Assistant, in 1960 an Associate Professor of Electrical Engineering at The George Washington University, Washington, D. C., where he was in charge of the computer facilities of the School of Engineering. He was also a staff member of the National Academy of Sciences—National Research Council project on the use of computers in biomedical sciences; consultant mathematician at the National Bureau of Standards, data processing systems division; and consultant to the National Library of Medicine. Currently he is President of the National Biomedical Research Foundation, Silver Spring, Md.



R. S. Ledley

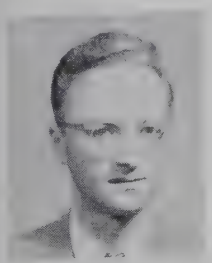
Dr. Ledley is a Fellow of the AAAS, and a member of the AIEE, American Physical Society, American Mathematical Society, Operations Research Society of America, and the Biophysical Society.

❖

Meier M. Lehman, for a photograph and biography, please see page 294 of the June, 1961, issue of these Transactions.

❖

Alan L. Leiner (M'54) was born in New York, N. Y. on February 10, 1914. He received the B.A. degree in mathematics and physics from Yale University, New Haven, Conn., in 1936, after which he studied astronomy at Harvard Graduate School, Cambridge, Mass.



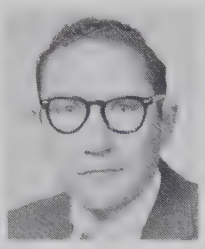
A. L. Leiner

From 1942 to 1944 he worked as an astronomer at the U. S. Naval Observatory in Washington, D. C. He then transferred to the National Brueau of Standards in the same city. At NBS he worked as a mathematical physicist on the radio proximity fuse until the end of the war, after which he turned to the computer field. As Chief of the Digital System Section at NBS, he directed the design of five large-scale systems, including the SEAC, DYSEAC, and the new PILOT. In addition, between 1954 and 1956, at the request of the Department of the Navy, he was lent by NBS to the Office of Naval Research, Washington, D. C. where he served as a consultant on computers. In 1960 he joined the IBM Research Center at Yorktown Heights, N. Y.

Mr. Leiner is a member of the ACM, the SIAM, the Washington Academy of Sciences, the AAAS and the AIEE Sub-Committee on Automated Design Techniques.

❖

Harry Loberman was born in Detroit, Mich., on October 27, 1924. He received the B.S. and M.S. degrees in physics from the University of Michigan, Ann Arbor, in 1946 and 1947 respectively, and a Teaching Certificate from Wayne State University, Detroit, in 1950. From 1947 to 1950 he was an instructor in physics at the Lawrence Institute of Technology, Detroit.



H. Loberman

From 1951 to 1956 he was employed by the Mine Fuze Branch of the Diamond Ordnance Fuze Laboratories, Washington, D. C. as project leader of the Special Devices Group. In 1956 he joined the Data Processing Division at the National Bureau of Standards, Washington, D. C. As a member of the Digital Systems Section he has participated in the logical design of the PILOT multiple computer system, including the formulation of methods to automate the production of the wiring plans and records for the PILOT. He is currently leading a team engaged in debugging the PILOT.

Mr. Loberman is a member of Phi Eta Sigma, Phi Kappa Phi, Phi Beta Kappa, and the ACM.

❖

Paul A. Mantek was born in Detroit, Mich., on January 30, 1920. He received the B.S. degree in electrical engineering in 1950 from the University of Michigan, Ann Arbor.

P. A. Mantek

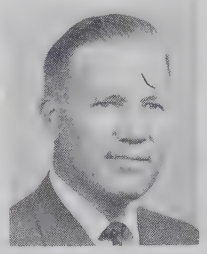
He joined the National Bureau of Standards, Washington, D. C., in 1949, where he worked on the design of input-output equipment for digital computers. He is now engaged in the application of high-speed semiconductor devices to digital computer circuits.

Mr. Mantek is a member of Eta Kappa Nu.

❖

Cecil B. Shelman was born in San Antonio, Tex., on September 30, 1922. He received the B.S. degree in electrical engineering from the University of Texas, Austin, in 1953. He did graduate work at Southern Methodist University, Dallas, Tex.

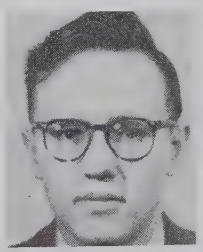
From 1953 to 1954 he worked in R&D laboratories of Phillips Petroleum Company, Bartlesville, Okla., on electronic instruments for process control. From 1954 to 1958 he was associated with Convair, Fort Worth, Tex. He did R&D work on magnetic-core logic circuits, obtaining two patents in that field. He developed the magnetic-core memory for the Cordic computer. Since 1958 he has been associated with Chance Vought Electronics, Arlington, Tex., where he has done circuit, logic and system design in digital computers, and where he was Project Engineer on Fingerprint computer development program.



C. B. Shelman

❖

Donald O. Smith was born in Santa Fe, N. M., in 1925. After attending the University of New Mexico, Albuquerque, N. M., for one year, he transferred to the Massachusetts Institute of Technology, Cambridge, Mass., and received the M.S. degree in electrical engineering from that Institute in 1951. Subsequently, he did graduate work in solid-state physics and received the Ph.D. degree from M.I.T. in 1955.

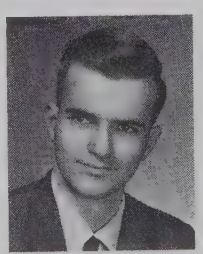


D. O. Smith

In 1955 he joined the M.I.T. Lincoln Laboratory, Lexington, Mass., where he has since been engaged in magnetic film work directed toward computer memory and componentry.

❖

Richard E. Stearns was born in Caldwell, N. J., on July 5, 1936. He received the B.A. degree from Carleton College, Northfield, Minn., in 1958, and the Ph.D. degree in mathematics from Princeton University, Princeton, N. J., in 1961. He was a National Science Fellow from 1958 to 1960 and a Teaching Assistant from 1960 to 1961.



R. E. Stearns

In July, 1961, he joined the General Electric Research Laboratory, Schenectady, N. Y., as a Research Mathematician in the Information Studies Section. He has done research in game theory and the structure of sequential machines.

Dr. Stearns is a member of the American Mathematical Society, the American Mathematical Association, Sigma Xi, and Delta Epsilon.

Edward H. Sussenguth, Jr. (M'58) was born in Holyoke, Mass., on October 10, 1932. He received the A.B. degree in applied science from Harvard University, Cambridge, Mass. in 1954 and the S.M. degree in electrical engineering from Massachusetts Institute of Technology, Cambridge, in 1959.

E. H. Sussenguth

From 1954 to 1957 he served as a Lieutenant in the U. S. Navy in the Far East. While studying at M.I.T. he was a Teaching Assistant and Research Assistant at the Research Laboratory of Electronics. Since 1959 he has been with the Machine Organization Department of the IBM Research Laboratory, Yorktown Heights, N. Y. He is now on educational leave of absence at Harvard University.

Mr. Sussenguth is a member of Sigma Xi.

❖

John V. Wait was born in Chicago, Ill., on October 1, 1932. He received the B.S.E.E. degree from the State University of Iowa, Iowa City, and the M.S.E.E. degree from the University of New Mexico, Albuquerque, in 1955 and 1959, respectively.

J. V. Wait

During his service with the Air Force from 1955 to 1957, he was a Research and Development Liaison Officer at the Air Force Special Weapons Center, Kirtland Air Force Base, N. M. His industrial experience includes summer employment with RCA Laboratories, Princeton, N. J., 1955, and the Nuclear Division, Kaman Aircraft Corporation, Albuquerque, N. M. in 1958 and 1959. He was a full-time Instructor in the Electrical Engineering Department at the University of New Mexico from 1957 to 1959. Since 1959 he has been employed as a Research Engineer with the Applied Research Laboratory, University of Arizona, Tucson, where he is working toward the Ph.D. degree in Electrical Engineering.

Mr. Wait is a member of Tau Beta Pi, Eta Kappa Nu, Omicron Delta Kappa, Pi Mu Epsilon, and Sigma Xi. He was holder of a NSF Fellowship for the 1960–1961 Academic Year at the University of Arizona.

❖

Arnold Weinberger (S'49–A'50–M'56) was born in Czechoslovakia on October 23, 1924. He received the B.E.E. degree from the College of the City of New York, New York, N. Y., in 1950 after serving in the armed forces. Afterwards he pursued graduate studies at the University of Maryland, College Park, Md.

A. Weinberger

In 1950 he joined the National Bureau of Standards' Electronic Computers Laboratory, Washington, D. C., where he participated in a succession of computer programs, notably SEAC, DYSEAC, and PILOT. On the SEAC, he carried out engineering work concerned with the expansion and operation of the machine. On the DYSEAC, he participated in the development of the logical design of the system, particularly the arithmetic circuitry and the detailed wiring plans from which the machine was constructed. Later he worked on logical design of novel high-speed arithmetic devices, such as adder and multiplier systems which were incorporated into PILOT. He also conducted research in graph theory to aid in design automation. As senior project leader on design automation he had major responsibility for the development of data-processing techniques for converting logical design data into detailed production plans and for the production of the detailed computer-generated fabrication plans and maintenance manuals for the PILOT. He has recently accepted a position with the International Business Machines Corporation Research Center, Yorktown Heights, N. Y.

Mr. Weinberger is a member of the ACM.

❖

James Bruce Wilson (S'56–A'57) was born on September 22, 1929, in Washington, D. C. He received the B.E.E. degree in 1958, from The George Washington University, Washington, D. C.

He was Research Associate at The George Washington University and is now Research Scientist and Secretary-Treasurer of the National Biomedical Research Foundation, Silver Spring, Md.

❖

Michael Yoeli was born in Piotrkow, Poland, on August 1, 1917. He received the M.S. degree in mathematics from the Hebrew University, Jerusalem, Israel, in 1957 and the D.Sc. degree in mathematics from the Technion, Haifa, Israel, in 1960.

M. Yoeli

He joined the Jerusalem Telephone Engineering Department in 1938. From 1948 to 1955 he was Head of the Telephone Switching Systems Planning Department at the Israeli Ministry of Posts, Jerusalem. Since 1955 he has been a Lecturer at the Electrical Engineering Department of the Technion. In 1960–1961 he was Visiting Assistant Professor at the Electrical Engineering Department, Syracuse University, Syracuse, N. Y., having been awarded a Fulbright Travel grant. For the summer of 1961 he joined the IBM Component Laboratory, Poughkeepsie, N. Y.

Dr. Yoeli is an Associate Member of the IEE and a member of the Mathematical Association of America.

Reviews of Books and Papers in the Computer Field

T. C. BARTEE, Review Editor

J. S. BOMBA, W. J. CADDEN, M. LEWIN, D. C. ENGLEBART, Asst. Review Editors

Please address your comments and suggestions to the review editor: Thomas C. Bartee, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass.

A. Sequential Switching Theory and Iterative Circuits

R61-118 On the State Assignment Problem for Sequential Machines. I—J. Hartmanis. (IRE Trans. on Electronic Computers, vol. EC-10, pp. 157–165; June, 1961.)

The author of this paper addresses himself to the notorious problem of coding the internal states of a given sequential circuit in such a manner that the complexity of the internal combinational network is reduced. The following assumptions are made:

1) The sequential circuit is clocked.
2) The present outputs are dependent only on the present internal state.
3) No "don't cares" are allowed in the state (or flow) table.
4) Unit delays are used for internal storage elements.
5) Two-level diode AND-OR logic is used to realize the combinational circuit.
6) No attempt is made to minimize the output logic.

Within this framework, the author develops a procedure for coding the states of those sequential machines which possess a property which he calls *partition with substitution property*. Briefly, a machine possesses a partition with substitution property if it is possible to divide its present internal states into several disjoint groups such that, given only the present inputs and the group to which the present internal state belongs, it is possible to determine, from the state table, the group to which the next internal state belongs. A set of rigorous procedures is given which allows one to find all such partition for a given state table, and the author shows how to use these partitions to obtain economical encodings. It is also indicated how such partitions can be used to merge a given state table.

It must be pointed out that the methods given are only applicable to a restricted class of sequential machines (namely those which possess the above-mentioned partitions). The procedures are fairly tedious to implement, and do not always result in absolutely minimal assignments. (This reviewer knows of at least one assignment for Machine C of this paper which requires 6 fewer diodes than the assignment given by the author.)

The most important contribution made by this paper consists of the fact that, to the best of this reviewer's knowledge, it contains the only results published to date which, in addition to giving an insight into the coding problem, also rest on a firm analytical basis. It is this reviewer's opinion that this paper points out a promising direction for research into the assignment problem, and that we need more general results of the type contained in this paper before sizable inroads can be made into this problem.

The exposition in this paper is quite good, and suffers only very slightly from the fact that the same set of symbols is used for both the present-state and next-state internal variables.

In summary, this paper is a clear exposition of some very elegant results which relate to the assignment problem. While the immediate practical application of these results is restricted, there is no doubt that they provide a very valuable insight into the assignment problem and that they point the way for further research in this area.

T. A. Dolotta
Bell Telephone Labs., Inc.
Murray Hill, N. J.

R61-119 Connective Properties Preserved in Minimal State Machines—Seymour Ginsburg. (*J. Assoc. Comp. Mach.*, vol. 7, pp. 311–325; October, 1960.)

The author investigates several properties of incompletely specified sequential machines, which are preserved in at least one minimum state machine of the given machine. Although he expresses the hope of ultimately using these properties to simplify state minimization procedures for incompletely specified machines, no such simplifications are given. This, however, may be a later aim of the author; and readers may find the properties of interest in other connections.

In a section entitled preliminaries, Ginsburg presents a flow table formulation of incompletely specified sequential machines. He calls the flow table a δ, λ-Matrix. The maps δ and λ describe, respectively, the next states and outputs for some of the present state and input pairs. The machines are called incompletely specified since δ and λ may be undefined for some state and input pairs.

In completely specified sequential machines an equivalence relation may be defined between states which give identical terminal behavior. This equivalence leads to a well-known state minimization procedure. For incompletely specified sequential machines this relation between states turns into a weaker relation where the transitive law no longer holds. Ginsburg defines this relation between two states p_1 and q_1 as follows: state $p_1 \leq q_1$ if for any input sequence $I_1, I_2, \cdots, I_k$ with the machine starting in state p_1 next states $\delta(p_i, I_i) = p_{i+1}$ exist for $1 \leq i < k$ and outputs $\lambda(p_i, I_i)$ exist for $1 \leq i \leq k$; then $\delta(q_i, I_i) = q_{i+1}$ exist for $1 \leq i < k$ and $\lambda(q_i, I_i) = \lambda(p_i, I_i)$ for $1 \leq i \leq k$.

A slightly more general relation for $p_1 \leq q_1$ can be defined in which $\lambda(q_i, I_i) = \lambda(p_i, I_i)$ where $\lambda(p_i, I_i)$ is defined, but $\lambda(p_i, I_i)$ need not be defined for all i. This later definition, used by Paull and Unger[1] leads to a more complete treatment when an output $\lambda(p, I)$ is not defined but a next state $\delta(p, I)$ is defined.

Ginsburg extends the $\leq$ relation to compare machines S and T, where $S \leq T$ if for each state p in S there is a state q in T such that $p \leq q$. A machine T is said to be a minimum state machine for S if (a) $S \leq T$ and (b) no other machine Y exists with fewer states than T such that $S \leq Y$.

Two restrictions on a machine S are imposed to obtain the connective properties for S:

1) For each state of S, there exists at least one input for which an output is specified, and
2) An output is specified for a given state and input whenever the next state is specified.

The three connective properties obtained by the author are then:

a) If S is a direct sum of strongly connected machines, then each minimum state machine of S contains a machine which is both a direct sum of strongly connected machines and is a minimum state machine for S.
b) If machine S has the following stability property—For some input I, whenever $\delta(q, I)$ is defined and there is a number k such that the next state after a sequence of k inputs I is again state q—then there is a minimum state machine W for S which has

[1] M. C. Paull and S. H. Unger, "Minimizing the number of states in incompletely specified sequential switching functions," IRE Trans. on Electronic Computers, vol. EC-8, pp. 356–367; September, 1959.

the property that from any state p of W, if $\delta(p, I)$ is defined then $\delta(p, I)$ equals the next state after a sequence of $k+1$ inputs I.

c) If two machines S and T are minimal with respect to each other, S and T are isomorphic.

If restriction 2) is removed, it is easy to devise examples for which the third result no longer holds. Thus, one wonders what the author meant in saying that, "These two assumptions lead to no real loss in generality and have the advantage of allowing simplifications in definitions, theorems, and proofs."

Several errors appear in the paper. On page 313 the end of the Paull-Unger Theorem should read that $S \leq T$, rather than that T "is a solution to problem Q for S," since the condition of minimality is not usually satisfied for all closed families of C-sets.

On page 315, Fig. 3 should have entries p_2, E^2 rather than p_2, E^1 for the entry in row p_2, column I^2.

In remark (1) on page 317 the author makes the statement that, "Lemma 2.1 cannot be changed to read: 'Let T be a solution to Problem Q for S. Then for each submachine W of T there exists a submachine V of S so that W is minimal with respect to V.'" This statement, however, is part of the conclusion of Lemma 2.1. The author probably meant that the Lemma could not be changed to read: "Let T be a solution to Problem Q for S. Then for each submachine V of S there exists a submachine W of T so that W is minimal with respect to V."

Finally, on page 323, Lemma 4.1, the word "stage" should be "state."

Raymond E. Miller
IBM Corp.
Yorktown Heights, N. Y.

B. Digital Computer Systems

R61-120

1) Design of an All-Magnetic Computing System: Part I—Circuit Design—H. D. Crane and E. K. Van De Riet. (IRE Trans. on Electronic Computers, vol. EC-10, pp. 207–220; June, 1961.)

2) Design of an All-Magnetic Computing System: Part II—Logical Design—H. D. Crane. (IRE Trans. on Electronic Computers, vol. EC-10, pp. 221–232; June, 1961.)

The MAD logic technique which was announced by H. D. Crane in 1957 depends only upon square loop magnetic material, conducting wire and pulse power supplies for the construction of complex computing circuits. The reclocking, reshaping and amplification of the digital signals are all provided by the ferrite and copper configuration entirely without the aid of conventional semiconductor or vacuum devices.

The present contributions describe the design and some of the operating experience on an experimental arithmetic unit consisting of 325 individual modules. Each module has two of the individual MAD logic elements.

The papers are valuable because the authors discuss frankly the limitations as well as the advantages of the computing technique. They describe the ways in which the capability of the individual module are derated in order to achieve reliable operation in complex systems. The experimental unit is sufficiently complex for an evaluation of the effects of the module characteristics on the system properties. The MAD logic module has been criticized because of its relatively low gain, its wiring complexity, its requirement of a complex and closely controlled power supply, and its high power consumption. The authors describe the techniques which were used to minimize these difficulties in the construction of a fairly complex subsystem.

The module developed has a fan-out of three and will perform an OR operation on two input signals. Any one of the three outputs may be inverted by modifying the output wiring. A ten phase clock source is used to drive the modules. The mechanical construction was arranged to minimize the wiring complexity and to minimize the coupling loop inductance. A clock pulse amplitude tolerance of ± 10 per cent was achieved with pulse durations compatible with a 20-kc operating speed.

The paper on the logical design includes a detailed description of the various parts of the arithmetic unit and the way in which these units cooperate. In addition, the author has included a good discussion of the different design philosophy required for the construction of magnetic logic computing systems in comparison to other, more conventional, systems of logic.

In brief, these papers provide an excellent description of the MAD all-magnetic logic system. Previous papers were intended only for device designers and circuit designers. However, in these publications there is material of direct interest to the logical designer and system designer who may wish to consider all-magnetic logic.

Edward P. Stabler
General Electric Co.
Syracuse, N. Y.

R61-121 Determination of Maximum Error of a Binary Multiplier—Yang Hsi-zeng. (*Automation and Remote Control [Automatika i Telemekhanika]*, vol. 21, pp. 709–713; February, 1961).

The paper is concerned with the operation of binary multipliers, of the type employed for digital integration in digital differential analyzers. The integration is carried out in the following manner: the integrand is stored in an n-digit register, where each digit is fed into an adder through a gate; the ith gate is controlled by a pulse generator whose frequency is proportional to 2^{-i}. The number of pulses at the output of the adder is then proportional to the desired integral. The author investigates the accuracy of this integration scheme, as influenced by the fluctuation of the output pulse frequency. Specifically, he derives a formula through which the maximum integration error can be evaluated for any specified n, and shows that the maximum integration error increases with n. The same formula can be readily extended to integration networks employing several integrators, such as used in the solution of simultaneous differential equations. Although the results in this paper are highly specialized, they may be of considerable value to those concerned with the design and use of digital differential analyzers.

Arthur Gill
University of California
Berkeley, Calif.

R61-122 Statistical Analysis of Certain Binary Division Algorithms—C. V. Freiman. (Proc. IRE, vol. 49, pp. 91–103; January, 1961.)

Division in most digital computers is executed as a sequence of iterative steps, which consist of additions or subtractions and left shifts. This method of division may be further classified according to the type of shifting which is employed. If the shifts of partial remainders are of fixed length, a constant number of quotient digits is generated during each iteration, and the division operation is completed in a fixed time. If shifts of variable length ($1, 2, \cdots, k$ digital positions) are allowed, the partial remainder is normalized during each iteration and k quotient digits are generated if the shift was k digital positions long. The number of iterations (and, consequently, the duration of the division operation) is variable. If fast execution of division is desired, the average number of shifts per iteration (*the shift average*) must be as large as possible.

The author of this paper presents a new, more rigorous, method for the calculation of the shift average for binary division with variable-length shifting. In order to determine the "steady-state" shift average, the formation of successive partial remainders is described in terms of a regular Markov chain. Steady-state probabilities for each interval of the partial remainder yield probability densities, from which steady-state shift averages are calculated. A general procedure for rational divisors in the range [0.5, 1.0) is described and illustrated by detailed examples. This use of the Markov chain model is a more rigorous approach than the calculation of "nominal" shift averages for an assumed uniform joint probability density of the divisor and successive partial remainders. A comparison between the nominal shift average, the steady-state shift average and the results of a simulation program of 2^{14} division problems shows that the nominal shift average is a useful approximation to the more rigorously and the experimentally obtained shift averages.

The author also discusses the increase of the shift average when two or three multiples of the divisor are available for the formation of the next partial remainder. (The divisor alone was available in preceding analysis.) The nominal shift average $\hat{s}$ is calculated as a reasonable approximation, and an increase from $\hat{s}=2.67$ (divisor only) to a maximum of $\hat{s}=4.05$ (divisor and two multiples) is observed. However, the discussion is quite brief and a more thorough evaluation of this method is certainly necessary.

The paper presents novel and useful insights into the statistical properties of the variable-shift division algorithm. The use of the Markov chain model is a welcome addition to the analysis of arithmetic processes, and the generous use of examples is highly com-

mendable. The introductory two-page section on the methods of binary division is a good tutorial article in its own right and makes the entire paper accessible to any interested reader.

Two items in Appendix II require correction: 1) The equation on line 8 should read "$\cdots$ of the form $\frac{1}{2}+2^{-k}(k=2, 3, \cdots)$ even though $\cdots$," that is, the symbol δ is unnecessary. 2) In Table IV, column "Discontinuities," the value "3−2D" in four lower rows should be replaced by "2−2D."

ALGIRDAS AVIZIENIS
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, Calif.

R61-123 A Parallel Arithmetic Unit Using a Saturated-Transistor Fast-Carry Circuit—T. Kilburn, D. G. B. Edwards, and D. Aspinall. (*Proc. IEE*, pt. B, vol. 107, pp. 573–584; November, 1960.)

The central theme of the paper is a fast-carry system using a method of fast-carry propagation through successive digit positions.

With mechanical switches or relays, a fast parallel adder can be obtained by letting the carry propagate conditions between two consecutive digit positions be represented by a closed contact. The carry propagate conditions between every pair of consecutive digit conditions can be set up simultaneously. Following that, the carry propagation through a series of closed contacts takes but a negligible length of time.

The electronic counterpart of a very short propagate time through a series of closed contacts is obtained by the authors using saturated transistors as closed contacts. It is based on the property of a saturated transistor that a pulse passes from emitter to collector with negligible delay if the pulse does not change the saturation conditions.

A 20-stage carry path constructed by the authors produced a total delay through the saturated transistors of 20 nsec of which about half was due to some 5 feet of wiring. This is in contrast to the 20 nsec of delay through a typical transistor gating stage.

The use of a fast carry system based upon fast carry circuits as opposed to logical means of speeding up the carry is not without its disadvantages. In the system described, the carry circuits require voltage levels different from those of the standard logical gating stages. Furthermore, to prevent excessive differences in current drain and voltage drop that confront the carry signal between the extremes of short and long carry propagation paths, an emitter follower is inserted after each group of six stages.

Standardization of circuits, of course, is worth a reasonable amount of additional hardware, provided that no substantial sacrifice in speed is entailed. The carry scheme devised by the authors, when projected to a 40-stage adder, appears to involve a carry path delay equivalent to between five and six standard gating stages. Considering the limited fan-in and fan-out of the standard gating stages, the carry speed compares favorably with carry schemes where speed is obtained through logical organization.

ARNOLD WEINBERGER
IBM Res. Ctr.
Yorktown Heights, N. Y.

R61-124 Binary Arithmetic—George W. Reitwiesner. (In "Advances in Computers," Franz J. Alt, Ed., Academic Press Inc., New York, N. Y., pp. 231–308; 1960.)

This chapter analyzes ways of efficiently performing arithmetic in a binary digital computer. The arguments are independent of specific logical structures, although the computer assumed as a basis for discusion is a machine of the Princeton class, with a parallel arithmetic unit in which additions (or subtractions) and shifts are the fundamental operations, and multiplications, divisions, and more complex operations are performed as sequential combinations of the fundamental operations. Numbers are assumed to lie in the range from -1 to $+1$ for most of the discussion; they are expressed in 2's complement representation.

The chapter starts with a description of the 2's complement number representation and the effects of scaling on the representation. The replacement, during arithmetic operations, of a number by its 2's complement, *i.e.*, by its residue modulo a power of 2, is then justified. This justification, though necessary, is usually omitted from discussions of binary arithmetic.

A detailed discussion of the algorithms for the fundamental arithmetic operations follows. A redundant binary representation using digits -1, 0, and $+1$, in which the number of non-zero digits is minimal, is then derived. This minimal representation is then used to reduce the number of additions or subtractions which must be performed in the compound operations. A detailed discussion of the multiplication, division, and square-rooting algorithms follows. An ultimate limit is derived for the number of additions and subtractions during division, and a similar argument is indicated for square-rooting. The chapter closes with comments on rounding, scaling, and multiple precision operations.

The material is always presented in a thorough fashion. Extensions of the arguments to other number ranges or other number representations, along the lines presented in this chapter, should be easy. The list of references provides a good cross section of activities in this area.

The only objection which one might raise is that the author perhaps tried to do too much: exhaustive attention to detail, a criticism anticipated in the author's introduction, frequently obscures the main issues, and an attempt to be general sometimes causes the discussion to become unnecessarily complicated, *e.g.*, in the simultaneous treatment of fractional and integral number representations.

This chapter is not intended for the uninitiated. However, with proper care, working through this chapter is a rewarding experience.

GERNOT METZE
Digital Computer Lab.
University of Illinois
Urbana, Ill.

R61-125 Analogue and Digital Computers—A. C. D. Haley and W. E. Scott, Eds. (The Philosophical Library, Inc., New York, N. Y.; 1960, 308 pages.)

This is a survey book, embodying a collection of chapters written by individual specialists on various topics relating mainly to analog and digital hardware.

There are ten chapters in this volume: an introduction; three chapters concerned with the operation, design, and circuits, respectively, of analog computers; and six chapters pertaining to number representation, operation, circuits, storage, input-output, and programming, respectively, of digital computers. The level of discussion in each chapter is mainly descriptive, vacuum tube circuits are predominantly used, and discussed on a very qualitative level.

This is not a volume for the active worker in either the software or the hardware end of the computer business. For even the general user it has major defects and omissions. It reflects the state of the art (primarily in England) in components and circuits as of the early fifties. It is an anthology that has no cohesive theme, viewpoint, or even a collection of tools to attack circuit, logical design or system problems.

It describes some of the techniques and elements that are in use without giving the user much of a feel as to the limitations and advantages of such use with respect to alternatives.

It is quite weak on logical design, and almost nonexistent in the area of over-all design considerations. A chapter on the systems aspect of computer design is thus a major omission.

In the area of usage and programming, once again the book is far behind the state of the art, coding being introduced and discussed in a single chapter at the machine language level, with almost no attempt to clarify or unify the relationship between usage and design.

At this stage of the development of the computer art, even an unsophisticated reader deserves to be shown "why" in addition to "what."

HERBERT M. TEAGER
Mass. Inst. Tech.
Cambridge, Mass.

C. CIRCUITS AND COMPONENTS

R61-126 Transistor Logic Circuits—Richard B. Hurley. (John Wiley and Sons, Inc., New York, N. Y.; 1961. 355 pp.+xvi pp. +1 Bibliography p.+5 Index pp. Illus. $10.00.

Mr. Hurley's latest book provides a complete single volume source for the designer of modern transistor logic. It is a book for engineers and physicists who want to build good equipment rather than for academic workers who have no equipment objectives. Mr. Hurley believes that there can be no separation between circuit design and the planning of logical combinations of these circuits.

"Transistor Logic Circuits" is a comprehensive and tutorial treatment of Boolean Algebra, component minimization techniques and semiconductor device circuit design theory. Only a knowledge of undergraduate engineering mathematics and network analysis are necessary tools for exploiting the carefully arranged information. This book is organized as a course text. The development of logic concepts is immediately succeeded by real circuit examples. The author has included electronic schematic diagrams often with specific part values and voltages. These circuits, while appearing specific, are always easy-to-understand examples of design philosophy, displaying the concepts developed in the text.

Other important examples of the good teaching techniques found in the book are the consistent and clear logic and circuit notations. Also, working terms are simply defined as they are introduced. I have found no instance of ambiguity anywhere in the work.

Little of the material appears to be new but rather the organization, presentation and comprehensive inclusion are unique. An exception might be in the sequential logic chapters where the techniques for state reduction and assignments are more lucidly presented than in any other source I have found.

Having worked intimately with several engineers who have studied the material in "Transistor Logic Circuits" at the University of California under Mr. Hurley, I have seen how effective integrated teaching techniques as are found in this book can be.

I want to recommend the book to any worker who wants to realize modern logical configurations with efficient reliable semiconductor circuitry.

Jerome A. G. Russell
University of California
Lawrence Radiation Lab.
Berkeley, Calif.

R61-127 Statistical Analysis of Logic Circuit Performance in Digital Systems—E. Nussbaum, E. A. Irland, and C. E. Young. (Proc. IRE, vol. 49, pp. 236–245; January, 1961.)

The authors present a technique for statistically analyzing the transient delays of computer switching circuits. The first part of the paper presents a brief review of two well-known techniques of statistical analysis, followed by the main body which treats a specific approach for analyzing the statistical distributions of delays of TRL circuits.

Using the Monte Carlo method, the authors have developed a computer program for this analysis. They have chosen a "propagation of delay model" to mathematically represent the dynamic behavior of the transistor for turn-on, turn-off and storage delay. This basic model is used to analyze any desired configuration of switching circuits.

To obtain the parameters of the equation, a Monte Carlo random number generator is used to select these values from the statistical distributions. These parameters are inserted in the equations of the mathematical model and the various delays are computed. The equation is solved iteratively, using the random number generator each time to select a new set of parameters, until a predetermined number of cases are computed. The various delays are then tabulated to give a complete distribution.

The single TRL gate was analyzed and distribution of delays compared with the computed worst case delays, pointing out the extreme pessimism associated with the worst case design philosophy. Comparisons were also made of delay distribution assuming both uniform and normal distributions for the parameters. As would be expected, the uniform distributions resulted in more pessimistic delays.

While the approach is not new, the authors have presented a fairly complete analysis and followed it with some interesting comparisons. A few questions are raised, however, by the reviewer, and the answers would be of interest to him.

1) In view of the analysis of both transitions plus saturation delays, it would be interesting to know to what accuracy the model represents the actual behavior of the circuit. The authors claim that inaccuracies of this model are much less than the spread of the expected distributions. The reviewer would be concerned to the extent that an inaccurate model may shift the mean value of delay. Some comparisons with actual circuit responses would certainly have been of value in pointing out the degree of deviation, if any, from actual performance.
2) The choice of the TRL circuit family puts a minimum burden on the accuracy of the mathematical model for a transistor. It would be interesting to know whether this analysis is extendible to widely used circuits such as diode-inverter logic.

R. J. Domenico
IBM Corp.
Poughkeepsie, N. Y.

R61-128 Contactless Semiconductor Switching Elements—E. V. Miller. (*Automation and Remote Control [Automatika i Telemekhanika]*, vol. 21, no. 17; February, 1961.)

The material in this paper is well presented and the analysis appears to be accurate. The subject is not new. Similar material has been discussed many times in western literature. The level of the material is approximately that of transistor circuit papers, presented in 1956–1957 dealing with basic transistor invertor stage design, *i.e.*, basic operation, temperature limitations, etc. The author also points out that diode transistor circuits offer basic advantages over all transistor circuits, etc. This of course has been well discussed in western lierature for over five years.

In short, the subject is not new but appears to have been new only to the author.

Richard H. Baker
Lincoln Lab., Mass. Inst. Tech.
Lexington, Mass.

R61-129 Semiconductor Devices and Applications—R. A. Greiner. (McGraw-Hill Book, Co., Inc., New York, N. Y.; 1961. 411 p.+76 Appendix pp.+5 Index pp.+xiv pp.)

This book might better be entitled "Transistors and Applications," since aside from very brief mentions of tunnel diodes, breakdown diodes and plate rectifiers, junction diodes and transistors receive all the attention. The book could be used for a junior- or senior-level university course on transistors. Problems are provided which well illustrate the topics discussed in the text.

Of the roughly 400 pages of text, about 100 pages deal with basic solid-state physics, some properties of germanium and silicon, and the theory of contacts. The treatment of such a vast area is necessarily cursory, and it is arguable whether much of the material contributes anything at all to the physical understanding of the circuit behavior of transistors. For example, some effort is expended in calculating the position of the Fermi level for both intrinsic and extrinsic semiconductors, but since the calculation depends on the density-of-states function which is presented with virtually no discussion, it is not clear that the student derives any benefit from this exercise. It will certainly not help him estimate the switching speed of a transistor in a regenerative circuit, to give an example of a topic which might have been more profitably discussed.

There are about 105 pages devoted to the physical theory of diodes and transistors, which end up in standard equivalent-circuit representations for large- and small-signal applications. High frequency and switching transistors are discussed later in the book in a separate chapter of 22 pages. The relations between the equivalent-circuit elements and the physical device parameters are generally pointed out, but the treatment of switching times is not illuminating. Following a qualitative discussion of charge storage, the standard rise storage and fall time equations are presented with very little indication of how they arise. The inverse alpha cutoff frequency is introduced in connection with storage time, but no definition or mention of the physical significance of this frequency could be found.

Turning to applications, one finds a very clear and complete treatment of biasing and stabilization problems (22 pages). Small-signal, power, and dc amplifiers occupy about 80 pages. Various gain and impedance formulas are developed for small-signal amplifiers in terms of the transistor hybrid parameters, and discussed principally for low frequencies where these parameters are real. The brief mention of amplifier response at high frequencies concludes that both voltage and current gain of a common-emitter stage must decrease above the beta cutoff frequency, *i.e.*, f_α/h_{fe}. This is incorrect, and arises from the uncritical use of a very restrictive equivalent circuit developed earlier.

There are two short chapters, totaling 30 pages, on regulated power supplies and sinusoidal oscillators. Finally, 40 pages are devoted to switching circuits. The treatment here is largely qualitative, no one circuit being treated in much detail. It is unfortunate that nothing is said about switching speed of the regenerative circuits since this can easily be done using material already developed or latent in the book.

A series of appendices provides information on matrix and parameter interrelationships, gain formulas, and manufacturer's data on a wide variety of transistors.

In summary, the reader of this book will learn something of the physical theory of transistors and how this relates to circuit performance, particularly in low-frequency applications. The writing is generally clear and readable. In the reviewer's opinion, however, there is too much physical theory which appears once and is never subsequently made use of. It is believed that a book in this area should contribute an over-all unity to the material discussed, which otherwise becomes simply a convenient collection of topics from the literature.

R. M. Scarlett
Stanford University
Stanford, Calif.

R61-130 ISABEL (Iso Status Accumulating Binaries using Extraordinary Logic)—J. A. Goss. (*Elec. Engrg.*, vol. 32, pp. 630–634; October, 1960.)

Mr. Goss's paper will be considered in four parts. The first part, an introduction, describes the fundamentals of decade counters constructed from binaries. The description contributes nothing new, but is clear and compact.

In the second part, the basic ISABEL concept is described. The concept utilizes parallel trigger inputs to flip-flops, through gates which are controlled by logic functions of previous flip-flops states. Rapid-carry counters of this type are not new. In a recent PGEC review,[1] Edson described several types of rapid-carry counters which have been known for quite a few years. Mr. Goss's contribution consists of a generalization of the type of counter which appears in Fig. 2 of Mr. Edson's review. The generalization is straightforward and entirely obvious—in hindsight. Although the result of such generalization can hardly be called "extraordinary logic," it does encompass a host of special counter configurations.

The design of these special configurations appears in the third part of the paper. Here Mr. Goss presents several codes for designing decade counters. A clever, semi-empirical trick employing Boolean algebra is described for simplifying the counter configuration; however, no definite design procedure for achieving the optimum is given. A severe limitation of the typical ISABEL codes is the impossibility of assigning a definite weighted value, such as the familiar 8-4-2-1, to each flip-flop. Consequently, decoding could be quite difficult.

The final part of the paper describes the flip-flop and gate circuits which were used to implement the logic design. The circuits are well known and are applied straightforwardly: nothing new or unusual appears in this section.

Despite her laboriously contrived name, ISABEL is really not the extraordinary new personality she would pretend. Beneath the glamour, she is really just another REBECCA (Routine logic Entailing Binaries Eliminating Cummulative Carry Advances).

A. K. Rapp
Res. Div.
Philco Corp.
Blue Bell, Pa.

R61-131 High Speed Scalers Using Tunnel Diodes—Philip Spiegel. (*Rev. Sci. Instr.*, vol. 31, p. 754; July, 1960.)

This paper is another illustration of the advantage of using tunnel diodes to perform a relatively complicated digital circuit function with a very simple configuration of elements. In this case, a scale-of-n counter is described which uses a series string of n matched tunnel diodes and two transistors.

The series string is biased with a current source giving $n+1$ stable states. It starts in the reset condition with all diodes in the low voltage state and each successive input pulse triggers one and only one diode to the high voltage state. The transistors are used to sense when the nth count has been reached and to reset the system.

In evaluating the circuit, one finds that the most critical design requirements are, first, having a sufficiently good match in the tunnel-diode characteristics and, second, controlling the input trigger pulse shape to insure that one and only one diode is triggered with each pulse. The author presents some numbers to show that one can, to a certain extent, trade input pulse requirements for more severe tolerances on tunnel diodes. Experimental work with circuits of this type indicates the tendency to skip one or more counts if the input pulse is too large or two wide. The reviewer feels that one cannot omit from the system an adequate limiting or pulse-shaping stage at the input. This would insure more reliable operation under typical laboratory conditions.

This paper is clear and concise and is recommended, with the reservations stated above, to anyone interested in a very simple, high-speed counter.[1]

Morton H. Lewin
RCA Labs.
Princeton, N. J.

R61-132 Linear Graphs and Electrical Networks—Sundaram Seshu and Myril B. Reed. (Addison-Wesley Publishing Co., Inc., Reading, Mass.; 1961. 315 pp.)

Linear graphs are playing an increasingly important role in modern system theory, particularly in that branch which deals with linear, lumped and time-invariant components. Traditionally, however, the approach of electrical engineers to system problems has been, with a few exceptions, dominated by Laplace transform theory and related topics. On the other hand every electrical engineer is accustomed, early in his career, to formulate a given problem by means of (circuit) diagrams. It is thus surprising that so little attention has been given to the study of circuit properties resulting primarily from the interconnection of the components themselves, with secondary regard to the voltage-current, etc., interrelations of the individual components. Graph theory, already formulated by mathematicians interested in topology, provides the necessary background to consider these aspects. In the last decade a number of electrical engineers, initially concerned with the formulation of Kirchhoff laws, as well as mathematicians seeking applications, seriously focused their attention in this area. As a result, several articles were published and one or two chapters on this subject appeared in a few introductory and advanced texts. This timely book by Seshu and Reed fulfills the need of bringing together under one cover both a coherent development of linear graph theory and a number of the important applications that are found in various branches of modern electrical science and are scattered throughout the literature. In this respect the book is a first in the English language. Its authors, both educators, are known for their research efforts in this area.

The book consists of ten short chapters. The first five chapters (about 100 pages) are devoted to the basic concepts and the development of purely abstract graph theoretic results. In these five chapters the notions of circuits, trees, cut-sets, oriented and nonoriented graphs, planar and dual graphs, are carefully presented. Algebraic concepts used for the study of linear graphs (rings, fields, linear vector spaces) are briefly introduced; however, matrix algebra is used extensively without explanation. At the end of Chapter V there is a brief section (five pages) which outlines the significant results on graphs. This section is most useful for references purposes, particularly since several of these results have appeared as problems in the text and could have been overlooked by the reader, and in view of the fact that the remaining five chapters in the text are concerned with applications of these results. This section is also novel in at least two respects: the results are classified in an orderly way, and in most cases the name of the person who first proved the results as well as the year it was done, is given. The effectiveness of this outline is somewhat lessened by the symbols which were used in the text and which are not explained here. Fortunately, most items state where, in the text, more information, as well as the pertinent proof, can be found.

The remaining five chapters, about 200 pages or two-thirds of the book, are concerned with applications of linear graph theory. Chapter VI deals with the role of graph theory in the formulation of network analysis problems. Chapter VII est blishes certain topological formulae for computing network functions of passive bilateral (with and without mutual inductance) as well as active and non-

[1] J. O. Edson, Review no. R61-51, IRE Trans. on Electronic Computers, vol. EC-10, pp. 301–302; June, 1961.
Mr. Edson cites Richards[2] as the source of his illustrated circuits.
[2] R. K. Richards, "Arithmetic Operations in Digital Computers," D. Van Nostrand Co., Inc., Princeton, N. J.; 1955.

[1] See also F. P. Heiman, "100-Mc Tunnel Diode Ring Counter," Proc, IRE. vol. 49, p. 1215; July, 1961.

bilateral circuits. Chapter VIII is concerned with the enumeration of natural frequencies of RLC networks "by inspection," certain questions connected with the transformerless realization of driving point passive functions, and the location of zeros of transmission of two-terminal-pair grounded RLC networks. Chapter IX takes up the application of graph theory to switching circuits (with emphasis on the relationship existing between electric circuits and contact networks), sequential machines (connection and transmission matrices) and a brief reference to logic networks (4 pages). Chapter X makes passing mention of the application of graph theory to several areas: communication networks, probabilistic nets, calculus of binary relations, axiomatics and others. One section of this chapter is devoted to the discussion of signal-flow-graphs (familiarity with the signal-flow-graphs is assumed). At the end of each chapter there is an adequate supply of problems, some rather challenging ones (the book contains altogether 200 problems). The appendix should be singled out as another novel feature of the book. It contains a collection of what the authors call research problems leading to term papers or theses. The problems are grouped according to the expected degree of difficulty. A definite aid to anyone undertaking such a task is the extensive bibliography of the text (200 references).

In the writing of this book the authors have succeeded in reflecting the interest and enthusiasm they have for their subject matter. They have made a conscientious effort to be precise and to provide rigorous proof in the style of the mathematicians and still not lose their humor as engineers concerned with such matters as physical networks which should not "go up in smoke." The coverage is good, although a few important applications of linear graphs are touched too lightly or altogether missing, *e.g.*, the application of signal-flow-graphs to analog computers. If the book is ever revised the recent contributions of Guillemin and Biorci and Civalleri on the realization of conductance matrices based on graph theoretic properties (sign matrix), which perhaps came too late for this volume, should be included. The authors are to be commended for their efforts. The book will significantly aid those interested in becoming familiar with linear graphs and their applications. It is indeed illuminating.

N. DeClaris
Cornell University
Ithaca, N. Y.

R61-133 Digital Computation Elements, Based on the Principle of Integrating Voltage Pulses—E. K. Yuferova. (*Automation and Remote Control* [*Avtomatika i Telemekhanika*]. vol. 21, pp. 1165–1172; August, 1960. English trans.)

Two digital devices, based upon the integrating properties of rectangular hysteresis loop magnetic cores, are described in this paper. The first is a preset pulse counter in which capabilities of up to 120 counts per stage are claimed for laboratory versions. The second is a three-input, binary adder which produces sum and carry outputs 20 μsec after the inputs have been received.

Both of the circuits described require partial switching of cores. In the counter circuit, transfer of flux from a fully switched "pulse-forming core" causes partial switching of an originally saturated "integrating core." When a predetermined number of pulses occurs, the integrating core reaches the opposite state of saturation, an output pulse is generated, and the integrating core is reset to the initial state. The pulse-forming core and the integrating core are coupled by means of a regenerative transistor switch. This switch derives its collector voltage from the pulse-forming core so that the flux transfer is made nearly independent of variations in the supply voltage.

Counters of this type have appeared in the literature before. The author cites Pittman[1] and Van Nice and Lyman.[2] A third which comes to mind is by Sands.[3] These counters exhibit economy of equipment and power, but suffer from a high rejection rate in the selection of cores and transistors.

Two forms of adder are presented in this article. In both, the three binary inputs cause an integrating core to be set to one of four possible states (positive and negative saturation, and two intermediate states). In the first , a sequence of four constant volt-second reset pulses produces a succession of output pulses from the integrating core whose number is equal to the number of input ONEs. The output pulses are gated to two conventional core circuts which compute the sum and carry bits. In the second, a current pulse resets the partially switched integrating core, inducing a voltage pulse whose amplitude depends upon the amount of flux being reversed. Voltage levels are separated by connecting windings of the integrating core to the inputs of three biased transistor amplifiers which in turn drive three output cores. Series combinations of the output voltages of these cores produce the sum and carry bits.

The adders are quite original but, in this reviewer's opinion, of questionable value. If great gains in speed and reductions in component count were realizable using these devices, then the unfortunate combination of biased transistor discriminators with cores operating in partial switching modes might be offset. This does not appear to be the case. Present levels of technology allow faster, more economical circuits to be constructed using properties of cores and transistors which depend less upon tight specifications, and so the devices presented have little to recommend them.

A. James Lincoln
Instrumentation Lab. M.I.T.
Cambridge, Mass.

[1] G. F. Pittman, "A high-accuracy static time delay device utilizing transistors," *Trans. AIEE*, pt. I (*Communications and Electronics*), March, 1955.
[2] R. J. Van Nice and R. C. Lyman, "A predetermined scaler utilizing transistors and magnetic cores," *Proc. Natl. Electronics Conf.*, October 3–5, 1955.
[3] E. A. Sands, "Magnetically Controlled Counters," 1957 IRE National Convention Record, pt. 4, p. 173.

D. Memories and Access Circuits

R61-134 High-Speed Light Output Signals from Electroluminescent Storage Systems—G. R. Hoffman, D. H. Smith, and D. C. Jeffreys. (The Institution of Electrical Engineers, Paper No. 3217M, pp. 599–605; February, 1960.)

The use of a matrix of small electroluminescent cells for the storage of digital information is described in this paper. Information in the form of light flashes from the selected cells is read out a bit at a time and detected by a photomultiplier tube. An opaque mask over the array constitutes the permanent or semipermanent store.

The authors give a clear discussion of the response of electroluminescent cells to voltage pulses, the limitation on read-out speed, and the limitations on the number of cells which can be utilized in the matrix.

The light flash from the electroluminescent cell produces in the output circuit of the photomultiplier a voltage pulse whose shape is not suitable for triggering computer type circuts. For this reason the pulse is integrated and strobed.

Although the output pulses may rise in a fraction of a microsecond read-out times are limited to about 25 to 30 μsec per bit because of the afterglow or slow decay of the phosphor. New electroluminescent phosphors may improve this, however.

The total number of cells in the matrix or the storage capacity is limited by the signal-to-noise or discrimination ratio. This ratio depends on the geometry (size and position of the cell, mask and photocathode of the photomultiplier) as well as electrical considerations. Similar to coincident current memories, the half selected elements produce the unwanted outputs. However, in the electroluminescent matrix this condition can be improved by applying auxiliary voltages to the unselected lines, *i.e.*, in addition to $+V/2$ and $-V/2$ to the selected row and column, respectively, $-V/x$ and $+V/x$ are applied to the unselected rows and columns. The method of optimizing x is treated.

Although theoretical calculations indicate that satisfactory discrimination ratios should be obtained with arrays as large as 64×64, measurements on experimental panels suggest that 32×32 is the maximum.

Another feature limiting the use of electroluminescence in memory applications is the phenomenon of build-up. If after a period of rest a cell is subjected to a repetitive waveform, the light output requires a few cycles to build up to a steady-state value. Similarly, if a cell is excited by irregularly spaced pulses of voltage, it is found that the light emitted at each pulse depends on the previous "history." For this reason, the authors limit their discussion to a continuous sequential selection memory in which each cell is selected once every operating cycle.

Possibly, because of the preliminary nature of the results, no consideration is given to changes in panel brightness with time and how this might affect the usable life of the matrix.

In view of the limitations mentioned, it is not clear where this type of memory is superior when compared to other forms of storage. However, the paper does list the following as possible advantages for using electroluminescent elements:

1) Each element may be made very small so that relatively large numbers of digits can be stored in a small area.
2) The method of construction is suitable for the production of cells which are very uniform, at a relatively low cost.
3) The read-out may be very rapid, and is obtained in the form of a pulse of light. This can be expected to simplify pick-up problems.

FRANK L. MCNAMARA
Lincoln Lab., Mass. Inst. Tech.
Lexington, Mass.

R61-135 A Digital Store With Very Short Read Time—T. Kilburn and R. L. Grimsdale. (*Proc. IEE*, vol. 107, Pt. B, pp. 567–572; November, 1960.)

The steadily increasing number of fixed and mechanically changeable matrix memories reported in an advanced state of development reflects their increasing importance in data-processing systems. As remarked by the authors of this paper, large capacity stores with access times in the microsecond and submicrosecond range can be manufactured at a small fraction of the cost of a core store of equal speed and capacity. The reasons are to be found in their simpler fabrication and access circuitry. The stores reported to date fall into two categories. In the first,[1–4] a 1 or 0 is recorded by the physical presence or absence of a linear or quasi-linear coupling impedance at the intersections of orthogonal word and digit lines comprising the matrix. In the second,[5] magnetic switching elements are used in place of the linear impedances and data is stored by the inhibition or not of switching. The faster response of the linear impedance gives the first class a higher speed capability, and the element itself may be simpler and the drive currents lower. On the other hand, the delayed peak output voltage from a switching element simplifies differentiation against noise transients, and its threshold makes possible simpler forms of accessing. These are among the points to be kept in mind when comparing designs for a particular application. This paper adds significantly to the stock of pertinent information in this growing technology.

The paper describes a high speed store that uses rods of linear ferrite at storage locations to produce inductive coupling between word and digit lines. The access time is 0.1 μsec and the cycle time is 0.2 μsec. Output voltages of a few millivolts are obtained using 50 ma word select pulses with 0.02 μsec rise times. A one to zero ratio of sixteen is quoted, although it is not clear whether or not this is a worst case figure. Practical considerations of construction, reduction of interactions between adjacent storage locations, and the design of suitable access circuitry are discussed in fair detail.

Two designs for use in the Ferranti Atlas computer are described, the first being a fixed store of 4096 words of 52 bits made very simply of a woven wire mesh for the word and digit line matrix, backed by modelling clay to hold the ferrite rods in place. The second store has 2048 words and is changeable. The ferrite rods in this case are free to slide within plastic tubes at the individual storage locations, and may be blown in or out for a 1 or a 0 by machine. The contents of the entire store can be changed in 55 sec.

The means of accessing the fixed store are described in some detail. The store is essentially subdivided into 16 units each containing 256 words. The 16 units have common word lines in series and common digit lines in parallel. Thus, address decoding is performed by a 1/256 selection of word lines and a 1/16 selection in each of the 52 sets of parallel digit lines. It would have been of interest to have been given the reasoning behind this particular organization. As Takahashi and Watanabe[2] have pointed out, one of the serious problems in a store using linear coupling elements is the noise coming through unselected elements. Therefore, the design and organization of suitable access circuitry is a critical consideration. A number of organizations are possible. For example, Takahashi and Watanabe use 4096 diodes and transformers to access the word lines of a capacitance-coupled fixed store of the same size and speed, while in the inductance-coupled metal card memory[3] selection is shared by input and output circuits. We may look forward to a fuller discussion of this point in future publications.

The paper provides an excellent introduction to this type of store but would have been strengthened by a thorough discussion of the extent to which noise transients are important. A discussion of the ferrite rod parameters and the degree of quality control necessary would also have been valuable. It is becoming apparent that most alternative designs of semipermanent stores can be reduced to forms involving simple printed circuit or weaving operations amenable to mass production. The fine details of fabrication and circuitry, therefore, play a significant part in determining final costs.

U. F. GIANOLA
Bell Telephone Labs., Inc.
Murray Hill, N. J.

[1] J. Van Goethem, *Proc. IEE*, vol. 108, pt. B, 1961.
[2] S. Takahashi and S. Watanabe, *Proc. Symp. on Large Capacity Memory Techniques*, Washington, D. C.; May, 1961.
[3] I. Endo and J. Yamato, *Proc. Symp. on Large Capacity Memory Techniques*, Washington, D. C.; May, 1961.
[4] D. H. MacPherson and R. K. York, "Semipermanent storage by capacitive coupling," IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 446–451; September, 1961.
[5] U. F. Gianola, *et al.*, *Proc. Symp. on Large Capacity Memory Techniques* Washington, D. C.; May, 1961.

E. PROGRAMMING AND NUMERICAL METHODS

R61-136 Computer Programming Fundamentals—H. D. Leeds and Gerald M. Weinberg. (McGraw-Hill Book Co., New York, N. Y.; 1961. 368 pp., $8.50.)

The authors of this book have presented a series of programming techniques which will prove invaluable to the beginning programmer and may be beneficial to the most experienced. The development of these techniques is done in a systematic way such that starting with basic, well-known facts each new topic follows logically from the preceding subject. While the organization of the material is different from that of most other books, it is very well suited to the authors' purposes. This book would be an excellent text for programming courses for it covers the topics which are fundamental to good programming from defining the program and flow charting through coding and debugging to the final production run. It teaches good habits from the start and uses them through all the examples in the text.

The first two chapters which describe a computer and its use are excellent for the person who is new to the field because difficult concepts are described clearly using simple examples familiar to everyone. The discussion of flow diagramming is excellent. The authors immediately establish good techniques in this area and demonstrate them throughout the book. They emphasize the importance of flow diagramming in defining a problem before trying to solve it which is something many programmers fail to do. The authors are to be commended for their emphasis on good flow diagramming techniques and their concern for full documentation of all programs. They develop an excellent method for tracing normal program sequence on the flow diagram, and the entire chapter defines a flow diagramming system which could be profitably applied by all programmers as well as installations which need an effective method of communication between programmers.

The frequently confusing concept of indexing is presented very clearly. A thorough and separate examination of counting is made thus simplifying the subsequent explanation of address modification. By introducing input-output in a previous chapter they have many opportunities to use looping instructions in realistic programs, manipulating files and records, for instance, on the basis of some count. In the chapters on subroutines, the authors again make a plea for thorough documentation and checkout of widely distributed routines. In addition, they hope that management will be more careful to evaluate proposed routines to avoid duplication of previously written programs and limit the distribution of routines to those which make a significant contribution to the field. This is another area where programming needs strong leadership, and it is encouraging to see it emerging. The authors also emphasize the need for more complete "automatic programming" systems and better operating systems which all programmers can profitably use.

A short section on advanced programming languages of the Fortran level might have been helpful since these languages are becoming increasingly important in programming. Such a description would also be valuable to anyone who is reading the book in an effort to understand the current literature.

In summary, this book systematically develops good programming skills, and the authors include programming hints wherever possible. Their frank discussion of the problems of errors, both human and machine, is refreshing in its practical approach to this unfortunately persistent facet of programming. By introducing a limited but representative instruction set, and using it repeatedly, the authors give the

reader a complete familiarity with these instructions. The reasons for choosing an existing machine as the example are good for, as stated, it does lend authenticity to the subject. However, they treat the machine in such a way that it does not limit the usefulness of their techniques to persons interested specifically in the IBM 7090. The reviewer highly recommends the use of this book as a text in courses whose goal is to produce efficient, skillful programmers.

Ann Ewing
IBM Corp.
Poughkeepsie, N. Y.

R61-137 On the Synthesis of Control Programs in Systems Incorporating a Digital Computer—P. F. Klubnikin. (*Automation and Remote Control*, vol. 21, pp. 1554–1559; November, 1960. Trans. in *Automation Express*, vol. 3; January, 1961.)

The author derives a class of difference equations for a control signal in a linear error-sampled feedback system. His stated purpose is to retain system stability and minimize computational volume. The z-transform approach is used.

In reading this article, one can only feel that the author is incompletely familiar with the general literature on sampled-data control systems. His treatment of this broad subject is inadequate in comparison with books and articles appearing several years previous to his, such as the work of Ragazzini and his colleagues.

The author adopts an over-all z-transfer function of the form

$$G_1z + G_2z^2 + \cdots + G_mz^m, \qquad z = e^{-pT},$$

in order to have a finite settling time for the system. The only suggestion made as to the values of the constants G_i is that they add to one for unit response to a unit step input or that the polynomial be divisible by $1-z$ to some power for other classes of inputs.

Starting from a system flow diagram originally proposed by Tou in 1957, the author derives the difference equations which will produce the desired transfer function. The resulting computational volume is claimed to be small, but the program structure, on which so much of computational volume and speed depend, is not discussed.

The author makes it a point to state that the difference equations must be stable if the system is to be stable despite the opinions of "certain authors" to the contrary. The certain authors are not identified, but their opinions are not generally held, in this country at least.

The behavior of an experimental system using an over-all transfer function

$$0.2z + 0.2z^2 + 0.2z^3 + 0.2z^4 + 0.2z^5$$

is described without going into detail.

One other point deserves mention. The article contains an equation, incorrectly stated, unfortunately, for the z-transform of the tandem combination of a plant and zero-order hold. It is a useful idea, and has not been noticed elsewhere by the reviewer.

In conclusion, this article does not appear to be particularly valuable in the light of contributions made in prior publications.

Albert Hopkins
Instrumentation Lab.
Mass. Inst. Tech.
Cambridge, Mass.

F. Analog Systems

R61-138 Survey and Classification of Multiplying Devices—A. A. Masalov. (*Automation and Remote Control* [*Avtomatika i Telemkhanika*], vol. 21, pp. 1000–1051; April, 1961.)

The author gives a review of all the commonly known electronic means of multiplication. The paper is largely tutorial in style since no design or circuit details are included. It does, however, contain analysis pertaining to the errors of some of the multiplying methods. From the content it appears that the majority of the multipliers have been constructed and tested by the author. Numerous numerical results concerning accuracy and bandwidth are quoted. However, no details beyond simple block diagrams are provided. A comparative table of multiplier characteristics is included. This covers the entire range from very accurate narrow-band to moderately accurate wide-band multipliers.

In the conclusion it is stated that an accurate broad-band multiplier cannot be realized utilizing any single principle. It is suggested that combinations of different methods could be used. However, these combined multipliers are not discussed.

Erik V. Bohn,
The University of
British Columbia
Vancouver, B. C.

Abstracts of Current Computer Literature

(THROUGH JULY, 1961)

These abstracts and the associated subject and author index were prepared on a commercial basis by Cambridge Communications Corp. under the direction of Dr. Geoffrey Knight, Jr., who also publishes the abstract journal "Solid State Abstracts," and the card services "Solid State Abstracts on Cards" and "Computer Abstracts on Cards."—*The Editor*

CONTENTS

0. GENERAL

1635
The State of Digital Computer Technology in Europe by N. M. Blachman (Sylvania); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 256–265; June, 1961.

A general survey of the state of the computer art in Europe and the Middle East is presented, with emphasis on the countries whose work is less known. Striking differences between the European and American computer scenes are the small number of people involved in Europe, the greater emphasis on wired-in subroutines, and the use of large numbers of bits in instruction codes.

Bionics Symposium Held in Dayton, Ohio, 13–15 September 1960—*see* 1700.

A Bibliographical Sketch of All-Magnetic Logic Schemes—*see* 1666.

Logic Circuits Using Square-Loop Magnetic Devices: A Survey—*see* 1662.

1636
Operational Compatibility of Systems-Conventions; *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 266–267; June, 1961.

A list of conventions proposed by the General Standards Committee of SHARE for programming systems with a view to minimizing machine set-up time is set forth. Suggestions proposed are that all programming systems be writen as subroutines, that input-output unit assignments be parameters of the calling sequence for a system call, that minimum supervisory routines be provided, and that all input-output functions be isolated and subroutinized.

1. LOGIC AND SWITCHING THEORY

A Straightforward Way of Generating All Boolean Functions of *N* Variables Using a Single Magnetic Circuit—*see* 1668.

Codes based on Switching Functions and their Application in Practical Coding Methods—*see* 1744.

1637
A Generalization of a Theorem of Quine for Simplifying Truth Functions by J. T. Chu (RCA); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 165–168; June, 1961.

A method of Quine for identifying the core prime implicants of a given truth function, without obtaining all its prime implicants, is generalized under the so-called "don't care" conditions. It is shown that the method is equivalent to, and sometimes an improvement of, a result of Roth. When all the prime implicants (under the don't care conditions) of a truth function are given, the method becomes a generalization of a result of Ghazala and is equivalent to another result of Roth. It is also pointed out that the method may be used, in a way similar to Roth's, for simplifying truth functions.

1638
Detection of Disjuncts of Switching Functions and Multi-Level Circuit Design by A. Mukhopadhyay (University of Calcutta); *J. Electronics Control*, vol. 10, pp. 45–55; January, 1961.

A method for the detection of component functions of fewer number of variables, the disjunction of which forms a given switching function, is described. The multilevel electronic circuit synthesized from the component functions yields better economy in several cases, with regard to the number of logical circuit elements, than when the same function is synthesized from the "minimal" two-stage forms.

1639
Synthesis of Threshold Logic Combinatorial Networks by L. Dadda (Electronic and Polytechnic Inst. of Milan); *Alta Frequenza*, vol. 30, pp. 224–231; March, 1961.

The problem of synthesizing arbitrarily assigned switching functions using only threshold elements is considered. Elementary threshold functions, *i.e.*, functions that can be implemented by a single threshold circuit, are first characterized for the cases of 2, 3, and 4 variables. Two methods are then illustrated for the synthesis of nonelementary functions, and the results for the primitive functions of the symmetry classes of 2 and 3 variables are given.

1640
Minimal Characterizing Experiments for Finite Memory Automata by J. E. Mezei (IBM); IRE TRANS., ON ELECTRONIC COMPUTERS (Correspondence), vol. EC-10, p. 288; June, 1961.

It is shown that the cyclic minimal experiments for the study of automata of memory M may be read off directly from the transition diagram of the Moore model for the maximal automation of memory M as the sequences of input symbols associated with the arrows of directed, minimal closed paths that traverse all the nodes.

1641
Symbolic Logic and Automata by R. McNaughton (University of Pennsylvania); *U. S. Govt. Res. Repts.*, vol. 35, p. 601(A); May 16, 1961. PB 171 548 (order from OTS $1.00).

Certain languages of symbolic logic for the description of the behavior of finite automata are presented. The conditions under which a formula of these languages describes the behavior of an automaton are discussed in detail and a synthesis algorithm which constructs a state graph for an automaton given a formula that describes the automaton is presented. Some unsolved problems concerning these languages are listed.

1642
On the State Assignment Problem for Sequential Machines. I. by J. Hartmanis (GE Res. Lab.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 157–165; June, 1961.

Determination of economical state assignments for finite-state sequential machines is studied. The fundamental idea is to make a selection such that each binary variable describing the new state depends on as few variables of the old state as possible. In general, variable assignments in which the dependence is reduced yield more economical implementation for the sequential machine than other assignments. The main tool used is the partition with the substitution property on the sets of states of a sequential machine. It is shown that the existence of assignments with reduced dependence is very closely connected with the existence of partitions with the substitution property on the set of states of the machine. It is shown how to determine these partitions for a given sequential machine and how they can be used to obtain assignments with reduced dependence.

2. DIGITAL COMPUTERS AND SYSTEMS

1643
The Design and Simulation of an Information Processing System by H. M. Gurk and J. Minker (RCA); *J. Assoc. Comp. Mach.*, vol. 8, pp. 260–270; April, 1961.

The development of a natural language processing system with high input rate and special requirements is described. These requirements include not only interpretation, storage, and retrieval of data but also logical processing, correlation, and combination of data to produce a modified body of information for retrieval and analysis. Simulation of parts of the system confirms its feasibility as well as the enormous programming effort and improved means of handling input and output facilities required. Much more sophisticated file organzation is also necessary to structure the information and yet allow for the integration of all related data.

1644
Irreversibility and Heat Generation in the Computing Process by R. Landauer (IBM); *IBM J. Res. & Dev.*, vol. 5, pp. 183–191; July, 1961.

It is argued that computing machines inevitably involve devices which perform logical functions that do not have a single-valued inverse. The logical irreversibility is associated with physical irreversibility and requires a minimal heat generation per machine cycle, typically of the order of kT for each irreversible function. This dissipation serves the purpose of standardizing signals and making them independent of their exact logical history. Two simple but representative models of bistable devices are subjected to a more detailed analysis of switching kinetics to yield the relationship between speed and energy dissipation, and to estimate the effects of errors induced by thermal fluctuations.

1645
Logic Structure Tables by H. N. Cantrell, J. King, and F. E. H. King (GE); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 272–275; June, 1961.

A set of rules for writing and using logic structure tables in logical design is explained by means of simple examples. The logic structure of a vending machine using two-dimensional tables is presented. The two-dimensional nature of the logic tables enables the full expression of both the sequential and parallel aspects of logic. Tables can be compiled directly by computer programs, thus eliminating the need for flow charting and hand coding.

1646
Graphical Manipulation Techniques Using the Lincoln TX-2 Computer by H. H. Loomis, Jr. (Lincoln Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 601 (A); May 16, 1961. PB 153 485 (order from LC mi$2.70, ph$4.80).

The results of an investigation into the use of a computer-controlled oscilloscope display for graphical manipulation, including symbol drawing and positioning of symbols to form a drawing, are reported. The structure and operation of a program to draw symbols and to use these symbols for the construction of a drawing are discussed. Finally, the type of program which would aid the designer in his logical circuit designing by preparing and filing the drawings and analyzing the circuits for such items as signal delays and unit loading factors is briefly considered.

1647
A Divisionless Method of Integer Conversion by W. K. Clarkson and B. M. Prince (Ramo Wooldridge); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 315–316; July, 1961.

A simple divisionless method for converting from base b_1 integers to base b_2 integers using base b_1 arithmetic is described. The method stops automatically when the last b_2 digit has been processed. It may be used for any bases and any computer word length. The method is to approximate $d=1/b_2$ in the base b_1 number system, form the Entire function $[Nd]$ (where N is the number being converted), and compute $r=N-[Nd]b_2$. If $r<b_2$, then the first digit $n_0=r$; but if $r\geq b_2$, then $n_0=r-b_2$. N may be processed iteratively digit by digit.

1648
Division and Square Root in the Quater Imaginary Number System by M. Nadler (Cie. des Machines Bull.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 192–193; April, 1961.

The quater imaginary system is one whose base is $(2i)$. Iterative methods for finding the reciprocal and the square root of a number expressed in this system are developed. The first procedure consists in finding a series of factors b_j such that $d\pi b_j\rightarrow 1$. Then $\pi b_j\rightarrow d^{-1}$. Similarly, if a series of factors b_j is such that $d\pi(b_j^2)\rightarrow 1$, then $\pi b_j\rightarrow d^{-1/2}$ and $d\pi b_j\rightarrow d^{1/2}$.

1649
Reducing Computing Time for Synchronous Binary Division by R. G. Saltman (Sperry Gyroscope); IRE Trans. on Electronic Computers, vol. EC-10, pp. 169–174; June, 1961.

A method of shortening the computing time for binary division by performing division radix 2^p on the binary operands, where p is a positive integer, is described. Each quotient digit radix 2^p is computed in almost the same time required to determine a binary quotient digit. Therefore, computing time is reduced by approximately the factor p over conventional binary division. The method is most useful for synchronous machines, but can be applied to either serial or parallel operation. The theory of nonrestoring division in any integral radix r is discussed. Each quotient digit is considered as the sum of two recursive variables a_k and b_k, whose values depend on the divisor multiplier and relative signs of the partial remainders. The divisor multiplier is limited to odd integers in order to determine the quotient digit unamibiguously. Using a_k and b_k, a single recursive equation combining all sign conditions is derived. This permits the derivation of the correct round-off procedure and shows that binary nonrestoring division is a particular case of nonrestoring division, radix r. An arrangement of components for a serial computer and a sample division for radix four are given.

1650
Programmed Error Correction on a Decimal Computer by G. M. Weinberg (IBM); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 174–175; April, 1961.

A generalization of programmed Hamming single-error correction to apply to a decimal machine is described. Check characters are aligned exactly as in the Hamming scheme, except that the check characters are now formed by taking the 10's complement of the modulo ten sum of the appropriate information digits. If, after transmission, the recalculated check sums are all zeros, the message is accepted as correct. If one check sum is nonzero, it has a single error. More than one equal nonzero check sums determine the size and position of the error. Multiple unequal nonzero check sums indicate a multiple error.

1651
Two-Dimensional Parity Checking by P. Calingaert (Harvard University); *J. Assoc. Comp. Mach.*, vol. 8, pp. 186–200; April, 1961.

Methods of error-correcting involving two-dimensional parity checking are discussed. The main area of practical application for these methods is magnetic tape systems. It is shown that the minimum distance D between two logical matrices is equal to d_rd_c, where d_r is the minimum distance between any two rows and d_c the minimum distance between any two columns of the matrices. Parallel decoding using row and column parities simultaneously avoids certain erroneous corrections, but has the disadvantages that no corrective action can be taken until all parity checks have been performed, and that correction depends in a complex manner on the individual checks. Serial decoding is simpler and methods of compensating for erroneous checking are described. Procedures for selecting array sizes that give a minimum redundancy for a given reliability are derived. Utilization of the two-dimensional structure permits the correction of nearly $d^2/4$ simultaneous errors with circuitry that individually corrects only about $d/2$ errors.

1652
Improvement of Electronic Computer Reliability Through the Use of Redundancy by J. Tierney and R. Wasserman (Hermes Electronics); *U. S. Govt. Res. Repts.*, vol. 35, p. 765(A); June 16, 1961. PB 154 087 (order from LC mi$3.00, ph$6.30).

Use of a "majority" type of redundant design to improve the reliability of complex digital systems is discussed. The systems considered operate under the following conditions: 1) The operating interval is sufficiently long or the system is so complex that the probability of successful operation is less than that called for by the application. 2) The system is unserviceable during the pertinent time interval for economic, physical, or strategic reasons. 3) Equipment cost and size are of secondary design importance (although order of magnitude increases are certainly undesirable).

1653
Application of Boolean Notation to the Maintenance of Switching Circuits by S. Alexander (English Elect.); *Electronic Engrg.*, vol. 33, pp. 372–374; June, 1961.

A method for the notation of switching circuits is proposed with special regard to its application in the drawing up of test and maintenance schedules. The system is based on Boolean algebra, which is already used extensively for the design of switching circuits. Several examples including a sample "test schedule" are given.

1654
The RCA 601 by K. Kezarsky and A. Mendelsohn (RCA); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 197–199; April, 1961.

The main operating characteristics of the RCA 601 system are described. Emanating from a central computer with 8192 56-bit words of 1.5-μsec storage are three channels —input-output, storage, and control. These channels provide standard interfaces to the devices with which they communicate, thus facilitating the addition of new modules. The central unit also provides decimal and binary arithmetic and means for handling a variety of character sizes. The status of all current input-output operations is maintained in the memory by tables accessible both to the programmer and to automatic controls. Flexible instruction length and housekeeping arrangements also contribute to system versatility.

1655
The Philips Computer PASCAL by H. J. Heijn and J. C. Selman (N. V. Philips); IRE Trans. on Electronic Computers, vol. EC-10, pp. 175–183; June, 1961.

PASCAL, a binary parallel computer with a word length of 42 bits, a clock-pulse repetition time of $1\frac{1}{2}$ μsec, performing, on the average, 60,000 operations/sec, is described. Wired-in floating-point facilities are provided. Core storage is backed by a drum and by magnetic tape. Modification versions for indexing as well as for stepping-up purposes are given. Special instructions include count and repeat instructions, jumps on the result of an earlier comparison, and two kinds of link instructions for facilitating the use of subroutines and interpretative programs. Transfer instructions enable a simultaneous bidirectional data flow between drum and cores or between tape and cores while computations are going on.

1656
The Evolution of Design in a Series of Computers, Leo I–III by J. M. M. Pinkerton (Leo Computers Ltd.); *Computer J.*, vol. 4, pp. 42–46; April, 1961.

The evolution of design philosophy in the Leo computer series from Leo I, a modified EDSAC with improved I/O facilities through Leo III, a large-scale parallel machine with multiprogramming, interrupts, independent I/O operations, and other features of modern large-scale data processors, is described. Leo I is a serial tube machine with small store (2048 words) and no index

registers. Leo III has a storage access time of 6 μsec, optional storage protection, checking of all operations except arithmetic, read-only memory controls, wired-in radix conversion instructions, and a high degree of system modularity.

A Table Look-Up Machine for Processing of Natural Languages—*see* 1693.

1657
Control Pulse Generation for a Digital Differential Analyser by P. L. Owen, M. F. Partridge, and T. R. H. Sizer (Royal Aircraft Estab.); *Electronic Engrg.*, vol. 33, pp. 364–371; June, 1961.

The generation of control pulses for a serially operated transistorized digital differential analyzer, CORSAIR, is discussed. Control pulses are needed in a serial digital computer to identify the word period and the digit period within a word, and to provide synchronized control signals. In CORSAIR, six discrete groups of pulses A, D, E, F, P, and p are needed. The A, E, and F pulses define the individual word and the D pulses define the increments. The frequency of the E pulses is 1/10 that of the P pulses, and the frequency of the F pulses is 1/5 that of the E pulses. The A pulses, which define the individual word and are designated A_1, A_2, $\cdots$, A_{50}, are formed by combining the appropriate E and F pulses. The P pulses define the individual digit period within each word and are designated P_1, P_2, $\cdots$, P_{20}. Each P pulse is further divided into 4 p pulses p_1, p_2, p_3, p_4, because the majority of the logical operations in the computer occurring within the duration of one P pulse fall into a four-part rhythm. There are: 1) clear destination, 2) transfer, 3) clear source, and 4) refill.

1658
Design and Applications of a Digital Differential Analyser by D. Lamb (Weapons Res. Estab.); *Proc. IRE Australia*, vol. 22, pp. 243–249; April, 1961.

The simulation of some control and physical systems requires greater accuracy than that offered by the operational units of an analog computer, and greater bandwidth than the speed of computation of most general-purpose digital computers permits. A suitable design for a digital differential analyzer (DDA) to fulfill these requirements of bandwidth and accuracy is considered. The variation of the accuracy of computation for a given word length with the nature of the transfer of information between integrators, and with the integration rule employed, is discussed. In particular, a simple method of ternary transfer of information, and a mode of integration which has not previously been employed, are described. Some applications to which the properties of the DDA are particularly suited are considered.

3. DEVICES AND BASIC LOGIC AND WAVEFORMING CIRCUITS

1659
Step-by-Step Design Techniques for Multilayer Thin-Film Networks by W. W. Carroll and F. F. Jenny (IBM); *Electronics*, vol. 34, pp. 90–93; May 19, 1961.

A technique of fabricating multilayer thin-film passive networks is described. The operational performance of these thin-film networks is determined and compared with that of conventional networks. Glass 0.020-in thick is used as substrate material. Aluminum is chosen for conductor, nichrome for resistor, and silicon monoxide for dielectric and insulation material. Wire leads are thermocompression bonded to the film-interconnecting tabs. The results of environmental tests of several hundred circuits are also included.

Cold-Cathode Tube Circuits for Automation—*see* 1789.

1660
Esaki Diode NOT-OR Logic Circuits by H. S. Yourke, S. A. Butler, and W. G. Strohm (IBM); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 183–190; June, 1961.

A basic technique for the development of Esaki diode NOT-OR logic circuits is presented. Two embodiments of the basic scheme which provide a logically complete system when combined with an OR-DELAY circuit are discussed. Emphasis is placed on the more economical of the two embodiments. A tolerance analysis which demonstrates that the technique enables the practical design of logic circuits is included. The requirements placed on Esaki diode characteristics and the speed limitations of the circuits are discussed. Examples of working circuits, including photographs of voltage wave shapes, are shown.

High-Speed Analog-to-Diginal Converters Utilizing Tunnel Diodes—*see* 1768.

1661
Millimicrosecond Digital Computer Logic by N. F. Moody and R. G. Harrison (Def. Res. Board of Canada); *Electronic Engrg.*, vol. 31, pp. 526–529; September, 1959.

A transistorized system of fast pulse logic that combines the efficiency of transformer coupled stages with digit delay tolerances approaching that of dc coupled systems is described. Logical circuits for OR, AND, INVERTOR, and RECLOCK and a driver which permits a "fan out" factor of 5 are described.

1662
Logic Circuits Using Square-Loop Magnetic Devices: A Survey by J. L. Haynes (Precision Instrument Co.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 191–203; June, 1961.

A capsule view of 24 square-loop magnetic logic circuits which have been proposed or developed so far, with a brief description of the way each circuit or circuit family meets the requirements of logic circuitry, is presented. All circuits are treated with a consistent terminology, and the generic relationships among circuits are stressed. Included in this survey are parallel and series transfer core-diode schemes, core-transistor schemes, and all-magnetic schemes of various topologies.

1663
The Core Amplifier, A Basic Circuit for Electronic Switching Equipment by H. Kok (N. V. Philips); *Philips Telecommun. Rev.*, vol. 22, pp. 81–93; January, 1961.

A core amplifier, consisting of a transistor and a ferrite core assembly, is described. The principle and design of this storage and switching element are discussed, and various applications in logical circuits such as counting circuits, code converters, and circuits for matrix control are considered. Finally, a method for the supervision of logical circuits made up of core amplifiers is discussed.

1664
Design of an All-Magnetic Computing System: Part I—Circuit Design by H. D. Crane and E. K. Van De Riet (Stanford Res. Inst.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 207–220; June, 1961.

The circuits used in a decimal arithmetic unit which utilizes ferrite magnetic elements and copper conductors only are described. The arithmetic operations of addition, subtraction, and multiplication are performed with a product and sum capacity of three decimal digits. The sole logical building block of this system is a two-input inclusive-OR module with a fan-out capability of three with any desired positive and negative combination. The system involves the use of some 325 modules, each of which contains two magnetic multiaperture devices (MAD's). The circuit and physical arrangement of the machine are described in detail. The system is controlled from a manual keyboard, and readout from the machine is via incandescent lamps controlled directly from the MAD elements, no intermediate elements being required. The "worst case" drive-pulse amplitude range for the completed machine, varying all clock pulses simultaneously, is ±10 per cent.

1665
Design of an All-Magnetic Computing System: Part II—Logical Design by H. D. Crane (Stanford Res. Inst.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 221–232; June, 1961.

A logical design technique for use with the particular module developed for an all-magnetic computing system is described. The detailed properties of this module, as well as the philosophy that led to its particular form, were covered in Part I (abstract 1664). Briefly, the module forms the (inclusive) OR function of two input variables. This function can subsequently be transmitted to three receivers, each transfer being independently logically positive or negative. The readouts are nondestructive and the transmitter module must be explicitly cleared before reading is again possible. In view of the relatively small fan-in and fan-out for this module, and since only the OR function can be directly formed during any single transfer, complex logic functions must be formed slowly, a step at a time. This step-by-step generation of functions results in the need for more modules than might otherwise be required, but aside from that, the synthesis techniques are not very different from those of customary logical design. In particular, the design of an arithmetic unit for decimal addition, subtraction and multiplication is outlined. Some comparisons between this particular all-magnetic logic scheme and conventional core-diode schemes are noted. Comparisons between magnetic

logic schemes in general and some other realization schemes such as ac-operated parametrons and conventional transistor systems are also made.

1666
A Bibliographical Sketch of All-Magnetic Logic Schemes by D. R. Bennion, H. D. Crane, and D. C. Engelbart (Stanford Res. Inst.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 203–206; June, 1961.

An all-magnetic logic scheme is one with which a workable digital system could be constructed involving only magnetic elements, current-carrying conductors, and sources of clock pulses. Historical developments of both resistance schemes (dependent upon coupling-loop resistance) and nonresistance schemes (possessing at least first-order independence of coupling-loop resistance) are described, with reference to all relevant published work known to the authors. Included are: 1) schemes using electric-circuit transfer linkage with simple cores, multipath cores, and thin-film elements; and 2) schemes using continuous magnetic structures in which transfer linkage is purely magnetic.

1667
Research on Multi-Aperture Magnetic Logic Devices by D. R. Bennion (Stanford Res. Inst.); *U. S. Govt. Res. Repts.*, vol. 35, p. 767(A); June 16, 1961. PB 148 231 (order from LC mi$5.40, ph$15.30).

Methods for describing and measuring the properties (static and dynamic) of magnetic materials, devices, and circuits are presented. A new approach to the development of practical switching models is introduced, for use in the analysis of flux transfer in all-magnetic circuits. Various means for realizing an OR-NOR logic module with one or more multiaperture devices, alone or in conjunction with toroids, are discussed. One such module makes use of a pair of PLUS-MINUS devices (components that can be wired for either direct or complementary transfer of information). A particular design for a device of this type was singled out for more detailed study. Ultrasonically fabricated units were tested individually and in operating circuits. Drive current ranges of 15–45 per cent were obtained for the various modes of operation. The device design is diagnosed and possibilities for substantial improvements are indicated. It is concluded that this particular approach to all-magnetic logic is a sound one.

1668
A Straightforward Way of Generating all Boolean Functions of N Variables Using a Single Magnetic Circuit by K. V. Mina and E. E. Newhall (Bell Labs.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 151–156; June, 1961.

A correspondence between the topologies of relay contact networks and magnetic circuits is applied to a relay tree to produce a magnetic structure capable of generating, in a simple manner, all Boolean functions of N variables. The basic magnetic topology may be distorted to achieve winding simplicity at the expense of magnetic circuit complexity. In the resulting arrangement, the drive, hold (variable), and reset windings are always in the same position, regardless of the function to be generated. Any one of the 2^{2N} functions of N variables is generated by linking a selected group of the output legs. The structure is such that all switching paths are of equal length, causing all outputs to be equal in amplitude. This balanced arrangement also permits the holding MMF to be significantly smaller than the drive MMF. The symmetrical holding scheme overcomes shuttle flux problems and reduces noise. An 8-leg manganese-magnesium-zinc ferrite structure operates easily at a 4-μsec cycle time with an output of 500 mv into 5 ohms. The peak-signal-to-peak-noise ratio is at worst 8:1. A 1-in-256 selector, using 17 16-leg structures, is under construction.

1669
Thin Film Magnetization Reversal Studies by R. L. Conger (Naval Ord. Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 601(A); May 16, 1961. PB 150 073 (order from LC mi$2.40, ph$3.30).

The suitability of very thin evaporated magnetic alloy films for use in coincident current memory devices in high-speed digital computers is shown. A method of preparing such films is outlined, and two methods of utilizing the films in memory devices are presented. The process of magnetization reversal in thin films is discussed.

1670
Optical Readout for Cryogenic Circuits by B. Rabinovici (IBM); *Rev. Sci. Instr.*, vol. 32, pp. 747–748; June, 1961.

An optical readout circuit which uses the Faraday rotation of the plane of polarization of a plane polarized light by a paramagnetic medium in a magnetic field to sense digital data from a superconducting circuit is described. An optically flat, cerium doped glass disk is subjected to a magnetic field by current pulses. A light beam (5400 Å) is directed through a polarizer onto the disk and reflected through an analyzer to strike a light detector circuit. The polarizer and analyzer are Glan-Thompson prisms set 90° out of phase with each other. The light detector is sensitive to any rotation of the plane of polarization caused by the disk. The rotatory power of the disk measured in the visible region at 1.8°K was 12°/100 Gauss mm.

Electroluminescent-Photoconductive Pattern Recognizer Organizes Itself—*see* 1706.

1671
Transistor Precision Pulse Shaper with Short Recovery Time by I. De Lotto (Centro Studi Nucleari); *Alta Frequenza*, vol. 30, pp. 219–223; March, 1961.

A pulse shaper with an RC timing network is discussed, and the influence of transistor characteristics on the performance of this circuit is analyzed. Some circuit modifications are suggested for obtaining: 1) a pulse-width precision better than 2 per cent without the need for individual adjustments, and 2) a pulse-width stability better than 1 per cent over a 20°C range of temperature. The circuit is so designed that the use of transistors having low base-emitter inverse voltage ratings (such as most of the faster types) is allowed.

1672
Reversible Decimal Counters by J. L. Goldberg (CSIRO, Austral.); *Electronic Tech.*, vol. 38, pp. 234–245; July, 1961.

Counters which reverse their direction of counting in response to external control signals are described. These devices are required in the application of optical interferometry to the precise measurement of length. A four-stage binary counter that skips states to yield a decimal count in either direction is considered in detail. A reversible bi-quinary counter realized by interposing transfer gates between a binary stage and a five-element ring counter is also discussed. Both types of counter are constructed from three basic circuit elements using junction transistors and diodes.

1673
Analysis of a Crossed-Film Cryotron Shift Register by H. H. Edwards, V. L. Newhouse, and J. W. Bremer (GE); IRE TRANS. ON ELECTRONIC COMPUTERS (Correspondence), vol. EC-10, pp. 285–287; June, 1961.

A crossed-film cryotron shift register mode requiring four loops per bit and two advance and two reset current sources is described. The criteria of operation are analyzed and a graphical solution for the equilibrium stored current is presented. The calculated highest speed of operation of the shift register corresponds to an information rate of just under 300 kc. Experimentally, the circuit can function perfectly at a rate of 140 kc.

4. STORAGE AND INPUT-OUTPUT

1674
Magnetic Cores, Characteristics and Applications by R. Stuart-Williams (Ampex Computer Products Co.); *Automatic Control*, vol. 15, pp. 37–43; July, 1961.

A general description of magnetic-core storage units is presented. The units are divided into general- and special-purpose classes, and information relative to both classes is given. Detailed descriptions of the operation of a typical buffer storage unit and a typical random-access storage unit are presented.

1675
Serial Matrix Storage Systems by M. Lehman (Israel Ministry Def.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 247–252; June, 1961.

Coincident-current techniques, usually associated with parallel ferrite-core stores, may also be used for the operation of serio-parallel or purely serial memories. Coincidence is established in the memory matrix between two currents representing an address signal and a time signal, respectively. The conditions under which such a store is economically justified are examined. It is shown how the properties of the time-controlled series store suggest adoption of a word-asynchronous design for serial digital computers, and facilitate the incorporation into small serial computers of autonomous transfers, automatic floating point operations, high-speed multiplication, division and shift orders, and asynchronous transfers between, e.g., a high-speed store and a magnetic drum.

1676
A Magnetic Associative Memory by J. R. Kiseda, H. E. Peterson, W. C. Seelback, and M. Teig (IBM); *IBM J. Res. & Dev.*, vol. 5, pp. 106–121; April, 1961.

The general principles of an associative memory in which data are identified by content, partial or complete, rather than address, are described. A magnetic-core realization of an associative memory in which data can be identified conventionally as well as associatively is explained. Memories as large as 4000 72-bit words with a 10-Mc search are feasible using the organization developed.

1677
Matrix Switch and Drive System for a Low-Cost Magnetic-Core Memory by W. A. Christopherson (IBM); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 238–246; June, 1961.

A system of ferrite-core matrix switches and drivers for a low-cost magnetic-core memory is reported. The memory uses coincident-current techniques and has a capacity of 10,000 characters with seven bits per character. A 20-μsec read-compute-write cycle features serial-by-character processing. Computing time is approximately 7 μsec, and read-write time is 13 μsec. The matrix switch requires only two sets of five drivers to select one out of 100 individual outputs. The drivers operate in an unusual three-out-of-five coding arrangement. A Set and a Reset driver, each using four transistors in parallel, are also required for the matrix switch. Two matrix switches provide the 200 X-Y half-select drives for a 100×100 seven-plane core array. At read time, two half-select current pulses of 250–300 ma are emitted with an effective 10–90 per cent rise time of 0.3 μsec. At write time, half-select current pulses with 1.2 μsec rise time are emitted on the same selected lines, but in the opposite direction. A new method of timing the drive current allows the read pulse to rise in 0.3 μsec, even with a low-voltage power supply and an inductive load that would otherwise limit the rise time to 0.8 μsec. All current-driving circuits use alloy, junction transistors. The drive current is furnished from a 10–12-v power supply, and temperature compensation of the drive currents is accomplished through control of power-supply voltage. Operating temperatures range from 10°C to 40°C.

1678
A 0.7 Microsecond Ferrite Core Memory by W. H. Rhodes, L. A. Russell, F. E. Sakalay, and R. M. Whalen (IBM); *IBM J. Rex. & Dev.*, vol. 5, pp. 174–182; July, 1961.

The design and performance of a low-power, high-speed magnetic-core memory are described. A two-dimensional array organization and partial switching of toroidal cores were employed. The drive system features a combination of a current-steering diode matrix and a load-sharing magnetic switch. The operating memory has a storage capacity of 73,728 bits and executes instructions reliably up to a repetition rate of 1.47 Mc. The discussion includes a description of the organization, the series-parallel delay line clock, the control of critical timing pulses, and the actual measured performance.

1679
A 2.18-Microsecond Megabit Core Storage Unit by C. A. Allen, G. D. Bruce, and E. D. Council (IBM); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 233–237; June, 1961.

A magnetic-core memory having a read-write cycle time of 2.18 μsec, an access time of 1 μsec, and a storage capacity of 1,179,648 bits is described. The array configuration and the design of the driving system are shown. The core and transistor requirements are discussed, and the sensing and the driving circuitry are described. Design factors which governed the choice of the three-dimensional system organization are presented.

Thin Film Magnetization Reversal Studies—*see* 1669.

1680
Static Reading of Magnetically Stored Digital Information by S. M. Rubens (Engrg. Res. Associates); *U. S. Govt. Res. Repts.*, vol. 35, p. 571(A); May 16, 1961. PB 154 747 (order from LC mi$3.00, ph$6.30).

It is demonstrated that the harmonically-variable reluctometer type of static reading device can be used satisfactorily to read magnetically-recorded binary digital information. With the circuits described, the device operates successfully and reliably for reading in contact records of 100 bits/in or less (recorded in contact) as long as the equipment is in continual readiness for use.

The Design and Simulation of an Information Processing System—*see* 1643.

1681
Panel Displays Real-Time or Programmed Data; *Electronics*, vol. 34, p. 68; May 26, 1961.

A display panel comprised of miniature bistable electromechanical reflective elements is described. The panel can be digitally controlled by real-time or programmed data. The basic elements of the display panel are reflectors which are mounted on pivots so that they can be driven between two stable positions. At one position, light from the source is reflected toward the viewer, and at the other position, it is reflected back toward the source. Thus, either white or black is displayed. These reflectors are mounted in a matrix similar to that used in computer core memories and are driven in the matrix by an X and Y coordinate drive circuit. Color displays are produced by operating the reflectors in groups of three representing the primary additive colors: red, blue and green.

Graphical Manipulation Techniques Using the Lincoln TX-2 Computer—*see* 1646.

5. PROGRAMMING AND CODING

1682
Problems Solved on High-Speed Computing Equipment of the Applied Mathematics Laboratory by H. Smith and L. Acton (David Taylor Model Basin); *U. S. Govt. Res. Repts.*, vol. 35, p. 765(A); June 16, 1961. PB 155 578 (order from LC mi$3.90, ph$10.80).

118 problems that have been programmed for solution on high-speed calculating equipment are described. These problems fall into two general categories: naval engineering and management data analysis.

Digital Computer Program for Superelevation Cam Design—*see* 1790.

Mathematical Analysis and Digital Computer Solution of Natural Frequencies and Normal Modes of Vibration for a Compound Isolation Mounting System—*see* 1760.

1683
Further Survey of Punched Card Codes by H. McG. Ross (Ferranti Ltd.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 182–183; April 1961.

A tabulation of the card codes of the leading European manufacturers is presented. In some codes, special features such as handling shillings and pence or representing non-printed accounting zeros are provided for. This table supplements a previous survey of American codes. ("Survey of Punched Card Codes," by Smith and Williams, *Commun. Assoc. Comp. Mach.*, p. 638; December, 1960.)

Design of an Improved Transmission Data Processing Code—*see* 1745.

On the Compilation of Subscripted Variables—*see* 1686.

Operational Compatibility of Systems-Conventions—*see* 1636.

1684
ALGOL 60 Confidential by D. E. Knuth (Calif. Inst. Tech.) and J. N. Merner (Burroughs Corp.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 268–272; June, 1961.

The greatly increased generality of ALGOL compared with earlier problem-oriented languages results in greater flexibility and also unsuspected traps. So many restrictions have been removed that technical details arise that are hard to learn and use correctly. Some of these more obscure features of ALGOL are considered and their usefulness is discussed.

The Use of Pegasus Autocode in Some Experimental Business Applications of Computers—*see* 1796.

1685
System Handling of Functional Operators by L. Lombardi (University of California); *J. Assoc. Comp. Mach.*, vol. 8, pp. 168–185; April, 1961.

The development of a programming language, in the form of an extended Fortran, that is capable of handling functional operators defined independently of their operands (*e.g.*, integrals whose integrand function is a dummy), is described. The process by which such operators are called is illustrated in several flow diagrams. An example of a library operator that provides for a programming procedure for evaluating an integral with a dummy integrand is given. Once the integrand is specified, the procedure is translatable into Fortran statements. In this way, the evaluation of functions which cannot be defined by simple arithmetic statements may easily be programmed.

1686
On the Compilation of Subscripted Variables by R. E. Nather (General Dynamics Corp.);

Commun. Assoc. Comp. Mach., vol. 4, pp. 169–170; April, 1961.

A method of evaluating subscripted variables by complete evaluation of the storage mapping function is described. Such a method is too time-consuming for a machine with built-in floating point. However, on a machine such as the RPC 4000, with programmed floating point, the time penalty is not prohibitive and the method results in the advantages that subscripted variables may have any number of dimensions, any arithmetic expression may be used as a subscript, and subscripting expressions may themselves contain subscripted variables.

1687
An Algorithm for Equivalence Declarations by B. W. Arden, B. A. Galler, and R. M. Graham (University of Michigan); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 310–314; July, 1961.)

An algorithm for handling declarations of equivalence of array names in algebraic compilers and for assigning the same storage locations to equivalent items is described. The strategy is to handle one equivalence class of array names at a time. Each equivalence class is built up by a chaining technique and assigned a storage area. The procedure is clarified by a specific example.

1688
Compiling Techniques for Algebraic Expressions by H. D. Huskey (University of California); *Computer J.*, vol. 4, pp. 10–19; April, 1961.

A compiling technique for translating from an algorithmic language to an object language in one pass by translating formulas from left to right is described in detail. Both source language and object language are highly systematized. Functions of only one variable are catered to, but the system is easily extended to cover multivariate functions. Subscripted variables and subroutines are handled in a natural and straightforward manner. Tables are used extensively.

1689
A Flexible and Inexpensive Method of Monitoring Program Execution in a Digital Computer by F. F. Tsui (Tech. Hochsch., Munich); IRE Trans. on Electronic Computers, vol. EC-10, pp. 253–259; June, 1961.

A method of monitoring the program execution in a digital computer on the basis of the flow diagram of the computing program is described. A comparatively low-cost equipment for monitoring a maximum of 64 boxes in a flow diagram has been constructed. The monitoring method can be used in connection with relative or symbolic addresses, compilers, etc. The user must provide only a flow diagram drawn on translucent paper in a certain form and the information to correlate this diagram with the computing program. A subroutine modifies the computing program as needed for the monitoring purposes and restores it to its original form when the user so desires. The monitoring introduces only a very small increase in computing time, requiring for each call-up of a box in the flow diagram a time amounting to only that needed for two simple unconditional jumps. The monitor can be used to present a visual dynamic picture of the progress of the program during the computation and to register, on occurrence, the whereabouts of an interruption, thus facilitating the tracing of the error.

Analysis of a Basic Queuing Problem Arising in Computer Systems—*see* 1741.

1690
Information Sorting in the Application of Electronic Digital Computers to Business Operations by H. H. Seward (Mass. Inst. Tech.); *U. S. Govt. Res. Repts.*, vol. 35, p. 765(A); June 16, 1961. PB 155 376 (order from LC mi$3.90, ph$10.80.)

Digital sorting and sorting by merging are examined. In the majority of cases, sorting may be achieved most economically with punched-card machines or magnetic-tape devices. In a system incorporating a general-purpose computing element, it is generally faster and more economical to use special-purpose sorting equipment. For applications involving large amounts of information and requiring rapid sorting, high-speed, high-density photographic storage may be practical. However, the processing time of the photographic medium should not be more than several seconds, and the information should be of such volume that the processing time is not significantly large in comparison with the total read-record time.

1691
Table Look-At Techniques by P. M. Sherman (Bell Telephone Labs.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 172–173; 175; April, 1961.

A table look-at technique, the basis of which consists of two steps, placing an argument in an index register and then operating on the contents of an address modified by that register, is described. The method compares favorably with table look-up in that only one reference to the table is made. However, empty words of storage corresponding to nonexistent values of the argument are interspersed in the table. Typical applications are integer conversion, multiple transfer of control, and certain types of sorting. The technique can be extended to tables of more than one dimension.

6. FORMAL AND NATURAL LANGUAGES, INFORMATION RETRIEVAL, AND HUMANITIES

1692
Atoms and Lists by P. M. Woodward and D. P. Jenkins (Royal Radar Estab.); *Computer J.*, vol. 4, pp. 47–53; April, 1961.

A straightforward exposition of McCarthy's LISP programming system together with motivation for its adoption is presented. The concepts of atoms and lists, subject language and metalanguage, machine format and recursion are described. Several simple programming examples including formal differentiation are supplied.

1693
A Table Look-Up Machine for Processing of Natural Languages by J. L. Craft, E. H. Goldman, and W. B. Strohm (IBM); *IBM J. Res. & Dev.*, vol. 5, pp. 192–203; July, 1961.

A table look-up machine based upon a photographic memory optimized for the processing of natural languages is described. It makes use of automatic retrieval of lexical information by means of novel addressing features which allow look-up of phrases regardless of length. Linguistic determinations made on the basis of the lexical information aready retrieved govern subsequent operations. In addition, a sentence buffer holds a sentence long enough to make the backward and forward passes which are required to make grammatical and contextual analyses.

1694
Table Look-Up Procedures in Language Processing, Part I, The Rax Text by G. W. King (IBM Res. Ctr.); *IBM J. Res. & Dev.*, vol. 5, pp. 86–92; April, 1961.

A method of addressing memories which is very powerful in the processing of natural languages, where the arithmetic or logical operations are either nonexistent or do not lend themselves to algorithmic description, is described. The main feature is the guarantee of initiation of an exhaustive search for a linguistic word at a point just beyond the desired address. Sequential search backwards not only locates an address if it is there, but also provides identification of a longest match first. The method is further extended to provide "conditional" addressing by prefixing subsequent addresses from information obtained in earlier searches.

1695
The Automation of General Semantics by F. W. Householder, Jr. and J. Lyons (Indiana University); *U. S. Govt. Res. Repts.*, vol. 35, p. 805(A); June 16, 1961. PB 153 805 (order from LC mi$2.40, ph$3.30).

The syntactic and semantic analysis of scientific English with the aid of an electronic computer is considered. The primary aim is the development of general mechanical routines for the reduction of complex sentences to their constituent simple sentences without loss of information-content. Work so far has been directed to: 1) the collection of a representative corpus of scientific writing in English; 2) key-punching and sorting the preliminary more limited corpus; 3) the elaboration of dictionary look-up and suffix-splitting routines for English; and 4) the mechanical determination of the syntactic function of words and of the boundaries of phrases and clauses.

1696
Machine Language Translation Devices System by J. F. McCarthy and J. F. McKenna, Jr. (Anderson-Nichols and Co., Boston); *U. S. Govt. Res. Repts.*, vol. 35, p. 805(A); June 16, 1961. PB 155 030 (order from LC mi$3.60, ph$9.30).

A complete coverage of the scope of the development program for the Machine Language Translation Devices System is presented. The coverage includes a complete phase breakdown of each of the tasks of the over-all development program including the development engineering for the Militarized High-Speed Page Printing System, the Code Conversion System, the Keyboard Input Units, and the results of the investigations for the Integration of the Magnetic Tape Units, the High-Speed Photoelectric Paper Type Reader, and the Militarized Digital Communications Kineplex System.

1697
Matching Inquiries for Index by M. A. Wright (Natl. Res. Dev. Corp., London); *Computer J.*, vol. 4, pp. 38–41; April, 1961.

A method for measuring the degree of matching between inquiries and index entries which allows for the effect of mistakes is described. A record is selected as the correct one if the degree of disagreement between its index and the inquiry is less than a preset threshold. If the threshold is exceeded, methods of determining the next index to be tested are described.

1698
Automatic Abstracting and Indexing—Survey and Recommendations by H. P. Edmundson and R. E. Wyllys (Planning Res. Corp.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 226–234; May, 1961.

In preparation for the widespread use of automatic scanners which will read documents and transmit their contents to other machines for analysis, a new concept in automatic analysis is presented—the relative-frequency approach to measuring the significance of words, word groups, and sentences. The relative-frequency approach is discussed in detail, as is its application to problems of automatic indexing and automatic abstracting. A summary of published automatic analysis studies is included. More sophisticated mathematical and linguistic techniques for the solution of problems of automatic analysis are also considered.

1699
An Indirect Chaining Method for Addressing on Secondary Keys by L. R. Johnson (IBM); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 218–222; May, 1961.

Methods for entering random-access files on the basis of one key are briefly surveyed. The widely used chaining method, based on a pseudo-random key transformation, is reviewed in more detail and an efficient generalization of the chaining method which permits recovery on additional keys is presented.

A Magnetic Associative Memory—*see* 1676.

7. BEHAVIORAL SCIENCE AND ARTIFICIAL INTELLIGENCE

1700
Bionics Symposium Held in Dayton, 13–15 September 1960 by J. C. Robinette (Wright-Patterson AFB); *U. S. Govt. Res. Repts.*, vol. 35, p. 765(A); June 16, 1961. PB 171 258 (order from OTS $6.00.)

The field of bionics is surveyed with special attention to the following subjects: 1) logic derived from the contemplation of neurons but applicable to the design of electronic networks of increased capacity and reliability; 2) theories, devices, and techniques based on or simulating visual and auditory perceptual processes; 3) the mechanization of higher functions, such as learning, self-programming, pattern recognition, decision making, and heuristic programs; and 4) the potential value of bionics, its present status, procedural methods and difficulties, and possible social consequences.

1701
On the Encoding of Arbitrary Geometric Configurations by H. Freeman (New York University); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 260–268; June, 1961.

A method which permits the encoding of arbitrary geometric configurations so as to facilitate their analysis and manipulation by means of a digital computer is described. Relatively simple numerical techniques can be used to determine whether a given arbitrary plane curve is open or closed, whether it is singly or multiply connected, and what area it encloses. Further, a given figure can be expanded, contracted, elongated, or rotated by an arbitrary amount. A number of ways of encoding arbitrary geometric curves to facilitate such manipulations, each having its own particular advantages and disadvantages, are treated. One method, the so-called rectangular-array encoding, is discussed in detail. In this method, the slope function is quantized into a set of eight standard slopes. This particular representation is one of the simplest and is most readily utilized with present-day computing and display equipment.

1702
A Note on Optimum Pattern Recognition Systems by W. H. Highleyman (Bell Telephone Labs.); IRE TRANS. ON ELECTRONIC COMPUTERS (Correspondence), vol. EC-10, pp. 287–288; June, 1961.

For a given rejection rate, the error rate in a recognition system is minimized if the following decision criterion is used: 1) choose pattern class k if

$$p_k F(V/a_k) \geq p_j F(V/a_j) \quad \text{for all} \quad j \neq k$$

and

$$p_k F(V/a_k) \geq \beta \sum_{i=1}^{c} p_i F(V/a_i), \qquad 0 \leq \beta \leq 1.$$

2) Reject the pattern if

$$p_j F(V/a_j) < \beta \sum_{i=1}^{c} p_i F(V/a_i) \quad \text{for all } 1 \leq i \leq c.$$

Here, p_i is the *a priori* probability of the occurrence of class i, $F(V/a_i)$ is the conditional probability of making the measurement V given that a number of class i is present, c is the number of pattern classes and β is a constant chosen to force the system to meet the given rejecton rate. In general, the proper value for β is determined empirically. However, in one important case, β may be determined analytically as $\beta=(\omega-\omega_0)/\omega$ where ω=cost of misrecognition, ω_0=cost of rejection, and the cost of recognition is assumed to be zero.

1703
The Design and Development of an (AN/APD-4 Film Editor) Readout Redundancy Reducer, QRC-60(T) by A. R. Bergen (Benson-Lehner); *U. S. Govt. Res. Repts.*, vol. 35, p. 764(A); June 16, 1961. PB 150 909 (order from LC mi$2.70, ph$4.80).

Design and development of a film editor used to simplify the recognition and annotation of trace data occurring at periodic intervals on 35-mm AN/APD-4 film, and to provide a visual readout of the time separation between two events is reported. Traces are annotated with an alpha-numeric or binary code of 175 combinations and can be edited at the rate of 15–30 points per minute. The film is advanced in incremental steps, the length of which can be set to correspond to the distance between traces. The error in the incremental step is less than 0.0025 in or one part in 2000, whichever is greater. A four-digit display provides a visual readout of the time separation between traces.

1704
Character Recognition and Photomemory Storage Devices Feasibility Study by D. M. Baumann, F. T. Brown, *et al.* (Mass. Inst. Tech.); *U. S. Govt. Res. Repts.*, vol. 35, p. 807(A); June 16, 1961. PB 147 787 (order from LC mi$4.50, ph$12.30).

A study undertaken to determine the feasibility of high-speed photoeletric scanning of printed material is reported. The desired goal is the development of a system that will scan and encode printed matter in a fashion suitable for digital computer input. Photoelectric character recognition, related optics, and photomemory development are discussed. A theory of weighted area scanning which yields an analog voltage by means of a photocell is presented.

1705
Computer Helps Make Machine for Blind; *Electronics*, vol. 34, pp. 26–27; July, 1961.

Use of a digital computer to help develop a reading machine for the blind is described. Photocells trigger reading-machine oscillators, generating a specific pitch, when darkened by a part of a letter. An IBM 650 is programmed to simulate the action of photocells in activating reading devices to find a photocell array that will generate the most distinctive combination of tones for each letter. Different types of photocell arrays can be investigated by programming the computer. To obtain interaction between the programmed letters and the programmed photocells, the letter is placed beneath a screen of 119 rows and 104 columns. A square in the screen is considered ON if a predetermined amount of it is covered by a part of a letter.

1706
Electroluminescent-Photoconductive Pattern Recognizer Organizes Itself by J. A. O'Connell (Genl. Tel. and Elect.); *Electronics*, vol. 34, pp. 54–57; July 14, 1961.

A sandwich module using flow table principles to recognize a 12-bit digital word is described. The device consists of a recognition panel, a storage panel and an input panel. The digital input is divided into four 3-bit words which are applied to successive recognition gates where they are compared with a pattern optically coupled-in from the storage panel. An ON condition is shifted from stage to stage when the digital information presented to the stage matches a pattern of binary words that it has been programmed to accept. Different recognition patterns are obtained by altering the pattern of illuminated controlling photoconductors in the flow table panel. Under single-shot operation, speeds of 3–4 msec per logical decision have been obtained. Under continuous operation, the speed is 5–10 times slower per decision element because of the slow photoconductor decay time.

1707
A Reader for Hand-Marked Documents by C. M. B. Reid (Leo Computers Ltd.); *Elec. Engrg.*, vol. 33, pp. 274–278; May, 1961

The design of a simple machine to read hand-marked documents such as invoices is described. The reason for undertaking the project is outlined, and problems inherent in sensing marks made by hand are pointed out. The mechanical and optical design of the reader is briefly described and a detailed account of the logical operations whereby the information on the document is sensed, staticized, processed, and punched into tape is given. Built-in alarms minimize the risk of mistakes due to creases or dirt; their usefulness is indicated in a summary of operational results.

Adaptive Servo Tracking—*see* 1778.

A Parameter-Perturbation Adaptive Control System—*see* 1777.

Transfer-Function Tracking and Adaptive Control Systems—*see* 1773.

1708
General Switching Theory by S. Litwin, R. McNaughton, and R. L. Wexelblat (University of Pennsylvania); *U. S. Govt. Res. Repts.*, vol. 35, p. 601(A); May 16, 1961. PB 171 555 (order from OTS $1.75).

Detailed reviews of a program to play GO and of work on languages of symbolic logic are given. The RENJYU program is described in detail, and a natural-language learning model is discussed.

1709
Adaptive Switching Circuits by B. Widrow and M. E. Hoff (Stanford University); *U. S. Govt. Res. Repts.*, vol. 35, p. 806(A); June 16, 1961. PB 150 227 (order from LC mi$300, ph$6.30).

An adaptive pattern classification machine (called "Adaline," for adaptive linear) devised to illustrate adaptive behavior and artificial learning is described. During a training phase, crude geometric patterns are fed to the machine by setting the toggle switches in a 4×4 input array. Setting another toggle switch tells the machine whether the desired output for the particular input pattern is +1 or −1. All input patterns are classified into two categories. The system learns a little from each pattern and accordingly experiences a design change. After training, the machine can be used to classify the original patterns and distorted versions of these patterns. At present, the purely mechanical adaption process is accomplished by manual potentiometer-setting. A means of automating this is being developed which makes use of multiaperture ferromagnetic devices. Solid-state adaptive logical elements will result that should ultimately be suitable for microminiaturization. Networks of such elements would be very effective in pattern recognition systems, information storage and retrieval-by-classification systems, and self-repairing logical and computing systems.

The Design and Simulation of an Information Processing System—*see* 1643.

8. MATHEMATICS

1710
Applications of the Complex Exponential Integral by M. S. Corrington (RCA); **Math. of Computation,** vol. 15, pp. 1–6; January, 1961.

A table of integrals in which definite integrals are evaluated in terms of the complex exponential integral is given. The Cauchy-Riemann equations are used to show how the present tables may be extended.

1711
"Direct Search" Solution of Numerical and Statistical Problems by R. Hooke and T. A. Jeeves (Westinghouse Res. Labs.); *J. Assoc. Comp. Mach.*, vol. 8, pp. 212–229; April, 1961.

"Direct Search" procedures for solving numerical or statistical problems not amenable to classical procedures are described. The typical problem is that of minimizing a function. A trial set of the dependent variables is chosen, and an attempt is made to improve the value of the function by altering the values of the variables. The method of choosing the new parameters is by pattern search. If a pattern of trials is successful, a new value of the function is obtained; if unsuccessful, a new pattern is attempted. The procedure is terminated when the step size of the pattern is sufficiently small to insure that the optimum has been closely approximated. Examples of the problems attacked are curve fitting and selection of an optimum function from a family of functions.

1712
A New Technique for Increasing the Flexibility of Recursive Least Squares Data Smoothing by N. Levine (Bell Labs.); *Bell Sys. Tech. J.*, vol. 40, pp. 821–840; May, 1961.

A method of performing recursive least-squares data smoothing in which optimum (or arbitrary) weights can be assigned to the observations is described. The usual restriction of a constant data interval can be removed without affecting the optimum weighting or recursive features. The method also provides an instantaneous (*i.e.*, real time) estimate of the statistical accuracy in the smoothed coordinates for a set of arbitrary data intervals. Optimum gate sizes for arbitrary predictions can be determined. Several applications are discussed.

1713
On the Approximation of Curves by Line Segments Using Dynamic Programming by R. Bellman (RAND Corp.); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 284; June, 1961.

A basic recurrence relation that permits convergence to the best line segment approximation to a function by iterative dynamic programming means is presented. The procedure may be extended to fitting segments of polynomials of any fixed degree to a given curve.

1714
Optimal Approximation for Functions Prescribed at Equally Spaced Points by H. F. Weinberger (University of Minnesota); *J. Res. NBS (Bull. Math. and Math. Phys.)*, vol. 65B, pp. 99–104; April/June, 1961.

Explicit upper and lower bounds for the value $F(u)$ of a linear functional F applied to a function $u(x)$ defined on the interval (0, 1) are given when u is prescribed at the $N+1$ points i/N, $i=0, \cdots, N$, and a bound for the integral of $u^{[k]2}$ is known. These bounds are optimal in the sense that they are attained for functions satisfying the prescribed conditions. Their computation requires the inversion of a matrix of order $k-1$ rather than of one of order N.

1715
An Iterative Method for the Inversion of Power Series by J. N. Bramhall (The Johns Hopkins University); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 317–318; July, 1961.

If a power series

$$y = \sum_i a_i x^i$$

is absolutely convergent in some neighborhood of the origin, then its inverse

$$x = \sum_i b_i y^i$$

may be calculated by writing the first expression as

$$x = y - \sum_{i=2} a_i x^i$$

and then iteratively using the second expression to substitute for x on the right-hand side. Test cases such as

$$y = \sum_n \frac{1}{n!} x^n$$

yielded

$$x = \sum_n (-1)^n \frac{1}{n} y^n$$

to seven-digit accuracy when run on the IBM 7090.

1716
Chebyshev Approximations of Some Transcendental Functions for Use in Digital Computing by A. J. W. Duijvestijn and A. J. Dekkers (N. V. Philips); *Philips Res. Repts.*, vol. 16, pp. 145–174; April, 1961.

Methods for obtaining best-fit truncated polynomial and continued-fraction approximations of continuous functions in a certain interval are described. Direct methods for obtaining approximations to best-fit approximations are also given. The choice of the interval of approximation is discussed, and refinement techniques for the polynomial approximation and for the truncated continued fractions are provided; 47 tables of polynomial approximations and best-fit continued fractions for specific functions are included.

1717
The Padé Approximant by G. A. Baker, Jr. and J. L. Gammel (Los Alamos Sci. Lab.); *J. Math. Anal. and Appl.*, vol. 2, pp. 21–30; February, 1961.

A method whereby the Padé approximant is used to calculate the asymptotic behavior of a function from the first terms of its power series is presented. The method is illustrated by two examples.

1718
On Approximating Transcendental Numbers by Continued Fractions by E. Karst (Brigham Young University); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 171; April, 1961.

Various continued fraction expressions for common transcendental numbers such as e^2 and π that converge more rapidly than the conventional continued fractions for such numbers are given.

1719
Coding Instructions for Floating Point Trigonometric, Inverse Trigonometric, Hyperbolic and Exponential Functions by B. A. Jensen (Lincoln Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 597(A); May 16, 1961. PB 153 052 (order from LC mi$1.80, ph$1.80).

Coding for the trigonometric, inverse trigonometric, exponential and hyperbolic functions in the floating point mode of the CG 24 computer is given. An accuracy of seven significant decimal figures can be achieved.

1720
A Further Note on Approximating e^x by D. Olivier (Carlton College); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 318; July, 1961.

Various rational approximations to e^x are obtained by taking successive approximants of a continued fraction expression for e^x. For example, the eighth approximant yields e^x to nine significant digits with only four divisions and one multiplication.

1721
Evaluation of the Incomplete Gamma Function of Imaginary Arguments by Chebyshev Polynomials by R. Barakat (ITEK); *Math. Comput.*, vol. 15, pp. 7–11; January, 1961.

Tchebycheff polynomials are used to evaluate incomplete gamma functions of imaginary argument. Tables of the evaluation are provided.

1722
Bessel Functions of Integral Order and Complex Argument by M. C. Gray (Bell Telephone Labs.); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 169; April, 1961.

An extension to the complex argument of a method due to Stegun and Abramowitz for generating Bessel functions of real argument is suggested. It is based on the addition formula

$$J_\nu(z+t) = \sum_{m=-\infty}^{\infty} J_{\nu-m}(t)T_m(z), \qquad |z| < (t).$$

Preliminary results indicate that the method compares favorably in speed and accuracy with methods using convergent or asymptotic series.

A Method for the Research of the Zeros of a Polynomial with an Analog Computer—*see* 1766.

1723
A Machine Method for Solving Polynomial Equations by D. H. Lehmer (University of California); *J. Assoc. Comp. Mach.*, vol. 8, pp. 151–162; April, 1961.

A machine method for solving polynomial equations that does not require special handling for multiple or closely spaced roots is described. An algorithm for determining whether a given polynomial has a root lying within a given circle in the complex plane is developed. By choosing two circles with the same center and differing radii, the location of the root can be isolated to within an annulus. The annulus may then be covered by smaller circles, the root isolated within one of these, and so on. Convergence is not at the quadratic rate of Newton's method, but rather at the geometric progression ratio of 5/12.

1724
The Effect of Parameters on the Roots of an Equation System by C. E. Maley (Carborundum Co.); *Computer J.*, vol. 4, pp. 62–63; April, 1961.

The effect of slight perturbations in the coefficients of a set of equations on the resulting roots is examined by means of a vector gradient method. In this way, the roots obtained may be evaluated and the effects of cancellation and round-off errors may be determined. By using the well-known relations between the coefficients and the roots of a polynomial, an iterative method of obtaining all the roots of a polynomial (real or complex) simultaneously is proposed.

1725
An Iterative Solution of Large-Scale Systems of Simultaneous Linear Equations by F. J. Bellar (Naval Postgraduate School); *J. Soc. Ind. Appl. Math.*, vol. 9, pp. 189–193; June, 1961.

A modification of Lanczyos' method of iteration that reduces the number of iterations to obtain the solution of $Ax=b$ is presented. Whereas Lanczyos obtains the best polynomial approximation to A^{-1} in the sense of absolute error, the method described gives the best polynomial approximation to A^{-1} in the sense of relative error.

1726
Eigenvalues of a Symmetric 3×3 Matrix by O. K. Smith (Space Technology Labs.); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 168; April, 1961.

An explicit solution for the eigenvalues of real symmetric 3×3 matrices is obtained by using Cardan's trigonometric solution of a transformation of the cubic equation $[A-\lambda I]=0$. The resulting method is computationally much simpler than using one of the standard eigenvalue methods for so small a problem.

1727
Truncations in the Method of Intermediate Problems for Lower Bounds to Eigenvalues by N. W. Bayley and D. W. Fox (Appl. Phys. Lab.); *J. Res. NBS (Bull. Math. and Math. Phys.)*, vol. 65B, pp. 105–112; April-June, 1961.

Two new procedures for determining lower bounds to the eigenvalues of linear operators are developed. The methods are based on the theory of semibounded self-adjoint operators in separable Hilbert space. Computation of the lower bounds is reduced to the solution of matrix problems. The procedures have immediate applications in the estimation of eigenvalues and eigenvectors of differential operators occurring in quantum mechanics.

1728
Solution of Tridiagonal Matrices by R. C. Wenrick (ACF Industries) and A. V. Houghton (University of New Mexico); *Commun. Assoc. Comp. Mach.*, vol. 4, p. 314; July, 1961.

Extremely simple recurrence relations for the solution of tridiagonal matrices are presented. The solution is obtained by successive back substitution.

1729
Comparison Theorems for Symmetric Functions of Characteristic Roots by M. Marcus, *J. Res. NBS (Bull. Math. and Math. Phys.)*, vol. 65B, pp. 113–116; April-June, 1961.

Necessary and sufficient conditions that $A-B$ is positive semidefinite Hermitian are proved. The conditions are in terms of the comparison of elementary symmetric functions of the eigenvalues of $A+X$ and $B+X$ as X varies over Hermitian positive definite matrices.

1730
A Remark on a Theorem of Lyapunov by O. Taussky (Calif. Inst. Tech.); *J. Math. Anal. and Appl.*, vol. 2, pp. 105–107; February, 1961.

It is shown that Lyapunov's theorem: if A is a real $n\times n$ matrix, then there exists a real positive definite matrix G such that $AG+GA'=-1$ if and only if A is stable, *i.e.*, $\mathrm{Re}[\lambda(A)]<0$ for all eigenvalues λ of A, is equivalent to the theorem: let $X=cI+K$, where $c<0$ and $K=-K'$ and K is a real $n\times n$ matrix. Let D be a real diagonal matrix. Then XD is stable if and only if all diagonal elements of D are positive.

1731
Solution of Systems of Ordinary and Partial Differential Equations by Quasi-Diagonal Matrices by M. A. Cayless (AEI Ltd.); *Computer J.*, vol. 4, pp. 54–60; April, 1961.

Methods of solving some rather complicated systems of ordinary and partial differential equations are described, including cases possessing eigenvalues and/or displaying nonlinearity. The equations are expressed in finite-difference form as "quasi-diagonal" matrix equations, and subroutines for solving these on the Ferranti Pegasus computer are given, including cases involving latent roots. The examples are from the theory of gas discharges, but the methods used are of general utility.

1732
Some Theoretical and Computational Matters Relating to Predictor-Corrector Methods of Numerical Integration by A. Ralston (Stevens Inst. Tech.); *Computer J.*, vol. 4, pp. 64–67; April, 1961.

Three matters relating to predictor-corrector methods are discussed. The first is a consideration of the replacement of classical predictors by direct extrapolation, smoothed and unsmoothed, of the first derivative. The second is the computational aspects of not iterating the corrector to convergence and the possible use of a convergence factor to test the corrector convergence. Finally, the desirability of using higher-order processes as opposed to modifying the results of lower-order processes is considered. Some numerical results are given to illustrate the topics discussed.

A Modified Lyapunov Method for Nonlinear Stability Analysis—*see* 1780.

1733
Successive Approximations and Computer Storage Problems in Ordinary Differential Equations by R. Bellman (RAND Corp.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 222–223; May, 1961.

A method for reducing the storage requirements of the quasi-linearization technique for solving nonlinear two-point boundary value differential equations is presented. The storage of the vector $x_{k-1}(t)$ in order to calculate $x_k(t)$ is avoided by transforming the boundary value problem in each iteration to an initial value problem and then recomputing the $x_i(t)$ as they are needed.

1734
A Practical Technique for the Determination of the Optimum Relaxation Factor of the Successive Over-Relaxation Method by H. E. Kulsrud (RCA); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 184–187; April, 1961.

A method, based on determining the spectral radius of a certain matrix required in the relaxation process, of estimating the relaxation factor for an over-relaxation method of solving elliptic partial differential equations is described. The estimate of the relaxation factor so obtained substantially reduces the number of iterations required for convergence.

1735
Some Numerical Experiments Using Newton's Method for Nonlinear Parabolic and Elliptic Boundary-Value Problems by R. Bellman, M. L. Juncosa, and R. Kalaba (RAND Corp.); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 187–191; April, 1961.

Using a generalization of Newton's method, a nonlinear parabolic equation of the form $u_t - u_{xx} = g(u)$, and a nonlinear elliptic equation $u_{xx} + u_{yy} = e^u$, are solved numerically. Comparison of these results with results obtained using the Picard iteration procedure shows that, in many cases, the quasi-linearization method offers substantial advantages in both time and accuracy.

1736
Topological Ordering of a List of Randomly-Numbered Elements of a Network by D. J. Lasser (Lockheed Aircraft); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 167–168; April, 1961.

A network of directed line segments free of circular elements is assumed. The lines are identified by their terminal nodes, and the nodes are assumed to be numbered by a non-topological system. Given a list of these lines in numerical order, a simple technique that can be used to create a list in topological order at high speed is described.

9. PROBABILITY, INFORMATION THEORY AND COMMUNICATION SYSTEMS

1737
Notes on a Pseudo-Random Number Generator by M. Greenberger (Mass. Inst. Tech.); *J. Assoc. Comp. Mach.*, vol. 8, pp. 163–167; April, 1961.

The influence of the parameter λ in the mixed congruential method $x_{i+1} \equiv \lambda x_i + c \pmod{2^P}$ for generating a sequence of pseudo-random numbers is discussed. It is shown that any $\lambda \equiv 1 \pmod 4$ leads to a sequence with a full period of 2^P numbers. The dependence of ρ, the serial correlation with lag 1, upon λ is considered. It is shown that a $\lambda = 2^{17}+1$ combined with $c \approx (1 \pm \sqrt{0.5})2^{34}$ yields a sequence with a period of over a billion and $|\rho|$ less than one billionth, properties which are adequate for most practical purposes. This choice of λ results in an efficient binary computer program; a new random number is generated by one shift and two add instructions.

1738
A Method for Evaluating the Area of the Normal Function by F. B. Baker (University of Maryland); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 224–225; May, 1961.

A method for calculating the area

$$A(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} \exp(-u^2/2)du$$

of the normal function is described. The abscissa of the normal curve is divided into 128 intervals whose increments are in octal units. The normal deviates corresponding to these intervals are entered in a table. A table look-up is performed to find an area corresponding to a deviate just less than the given argument. The remaining small area ΔA is then evaluated by a definite integration routine.

1739
Optimization Problems: Solution by an Analogue Computer by A. W. O. Firth (Redifon Ltd.); *Computer J.*, vol. 4, pp. 68–72; April, 1961.

Optimization problem solution on an analog computer in general is discussed and the field of linear programming is handled in detail. A method of solving linear programs that requires one amplifier per parameter and two amplifiers per constraint is presented. For large problems, a solution correct to 1 per cent accuracy converges in less than 20 sec. This is adequate for many applications where input coefficients are not known to an accuracy greater than 1 per cent.

1740
A Mean-Weighted Square-Error Criterion for Optimum Filtering of Nonstationary Random Processes by G. J. Murphy and K. Sahara (Northwestern University); IRE Trans. on Automatic Control, vol. AC-6, pp. 211–216; May, 1961.

A procedure for use in the design of a physically realizable time-invariant linear system for optimum filtering of a nonstationary random process in the presence of nonstationary random noise is presented. The criterion used to measure system performance is a mean-weighted square error. It is shown that the use of this generalization of the mean square-error criterion leads to a generalization of the Wiener-Hopf integral equation. A technique for solving this modified Wiener-Hopf integral equation is presented, and the application of the theory is illustrated in an example of the optimum synthesis of a missile interception system.

1741
Analysis of a Basic Queuing Problem Arising in Computer Systems by P. E. Boudreau and M. Kac (IBM Prod. Dev. Lab.); *IBM J. Res. & Dev.*, vol. 5, pp. 132–140; April, 1961.

A model which describes a basic junction or queuing structure, arising in a general computing system is subjected to a mathematical analysis. The results consist of several formulas describing the performance of various parts of the system. The feasibility of analyzing general queuing problems in this manner is stated, together with the results of a Monte Carlo simulation used for comparison purposes.

1742
Errors in Detection of RF Pulses Embedded in Time Crosstalk, Frequency Crosstalk, and Noise by E. A. Marcatili (Bell Telephone Labs.); *Bell Sys. Tech. J.*, vol. 40, pp. 921–950; May, 1961.

The probability of error in the detection of RF pulses embedded in a combination of Gaussian noise, time crosstalk from the tails of two neighboring pulses, and frequency crosstalk from an adjacent channel is calculated. It is shown that for a given probability of error, it is possible to maximize the pulse repetition frequency and simultaneously to minimize the channel spacing and signal-to-thermal noise by operating the system at a signal-to-thermal noise level close to the level of the combined time and frequency crosstalk.

1743
Determination of R_{cutoff} for Asymmetric Channel by B. Reiffen (Lincoln Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 601(A); May 16, 1961. PB 153 108 (order from LC mi$2.40, ph$3.30).

Previously, the author defined a computation cutoff rate, R_{cutoff}, for a discrete memoryless channel symmetric at its output with equally likely inputs. For information rates $R < _{cutoff}$, sequential decoding may proceed with an average number of decoding computations which does not grow exponentially with constraint length n. In this report, R_{cutoff} is defined for the general nondegenerate discrete memoryless channel with arbitrary input probabilities.

1744
Codes Based on Switching Functions and their Application in Practical Coding Methods by P. G. Neumann (Bell Telephone Labs.); *Nachrichtentech. Z.*, vol. 14, p. 254; May, 1961.

A class of codes for which the disadvantages of optimal codes can be avoided is developed. A code of this class is defined by means of a switching function whose variables are the digits of the code words and whose functional value serves to designate the ends of code words. An algorithm by which all possible functions, and hence codes, can be obtained is given. Two methods for attaining near-optimal average codeword lengths are given and illustrated with examples.

Carrier Phase Reversal Transmits Digital Data Over Telephone Lines—*see* 1787.

1745
Design of an Improved Transmission Data Processing Code by R. W. Bemer, H. J. Smith, Jr., and F. A. Williams, Jr. (IBM); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 212–217, 225; May, 1961.

An improved code capable of meeting the requirements of both communication and

data-processing needs is outlined. A major obstacle to standardization is the effect of any change on the collating sequences of established codes. The relative positions of the alphabet and the digits and the ability of codes of 6 or more bits to be mapped into a 5-bit Baudot-type code are important considerations.

1746
Data Transmission and the New Outlook for the Computer Field by M. V. Wilkes (Cambridge University); *Computer J.*, vol. 4, pp. 1–9; April, 1961.

Rates of error in telephonic communication of data and means of improving them are discussed. The standard methods of error detection and correction, such as one- and two-dimensional parity, Hamming and fixed weight codes, are reviewed, and the importance of data transmission for the future application of computers is emphasized.

1747
Minimum Polarized Distance Codes by M. P. Marcus (IBM); *IBM J. Res. & Dev.*, vol. 5, pp. 241–248; July, 1961.

Error detecting and correcting codes are discussed. It is shown that, for a given number of bits per character and a given minimum distance, the probability of undetected error in an asymmetric channel may be reduced from that in a symmetric channel by many orders of magnitude merely by the proper selection of coded characters. For a given minimum distance, an optimum selection of characters requires, as nearly as possible, the same number of "one" and "zero" bit failures to change one character to another. The concept of polarized distance is introduced, and it is shown that the probability of undetected error is related to the minimum polarized distance in both symmetric and asymmetric channels.

1748
The Use of Group Codes in Error Detection and Message Retransmission by W. R. Cowell (Bell Labs.); IRE TRANS. ON INFORMATION THEORY, vol. IT-7, pp. 168–171; July, 1961.

Group codes whose function is divided between error correction and error detection with retransmission are considered. For a given code, the minimum error probability is obtained if retransmission occurs whenever an error is detected. The redundancy added by retransmission is estimated, and the behavior of retransmission channels as the length of the code words increases is studied. Most of the analysis is applicable for the binary symmetric channel only, although some of the results apply to more general channels.

1749
A Direct Digital Method of Power Spectrum Estimation by P. D. Welch (IBM Res. Ctr.); *IBM J. Res. & Dev.*, vol. 5, pp. 141–156; April, 1961.

A method of digital power spectrum estimation involving the direct combination of the sample time function with sines and cosines is discussed. This treatment is in contrast to the normal indirect digital method which proceeds through the intermediary of the autocovariance function. All the practical design details necessary for the planning of a spectral estimation program are treated.

1750
Time and Frequency Crosstalk in Pulse-Modulated Systems by E. A. Marcatili (Bell Telephone Labs.); *Bell Sys. Tech. J.*, vol. 40, pp. 951–970; May, 1961.

The time and frequency crosstalk between gaussian RF pulses sent via adjacent frequency channels over the same transmission medium is calculated. Shapes of the transfer characteristics of the transmitting and receiving filters vary from gaussian to approximately that of a third-order maximally flat filter. The result permits design of transmitting and receiving transfer characteristics of adjacent PCM channels in such a way that the product of pulse spacing and channel spacing is minimized.

Adaptive Servo Tracking—*see 1778.*

1751
Wallops Island Preliminary Processing Computer Programs by S. M. Ornstein and R. J. Saliga (Lincoln Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 767(A); June 16, 1961. PB 154 587 (order from LC mi$2.70, ph$4.80.)

A portion of the data-processing system used to reduce instrumentation data from Wallops tracking radar is described. Data from three acquisition systems (primary, backup, and teletype) are processed by the 7090 Computer so as to produce a standard format IBM magnetic tape. This tape is then available for further processing. In addition, two preliminary output systems (for printed output and plotted output) that employ the standard format IBM tape as input and employ other programs to obtain printouts and plots are described.

1752
Error Statistics and Coding for Binary Transmission over Telephone Circuits by A. B. Fontaine (University of Wisconsin) and R. G. Gallager (Lincoln Lab.); PROC. IRE, vol. 49, pp. 1059–1065; June, 1961.

About 2000 hours of noise data on various digital data telephone links are analyzed with respect to the applicability of coding theory to such links. The analysis indicates that coding for error correction is inefficient for such systems, but that coding for error detection, along with feedback, is remarkably simple and effective.

Present Address Selection for DIGICOM, a Digital Communication System—*see 1788.*

10. SCIENCE, ENGINEERING AND MEDICINE

1753
Direct Cycle Nuclear Power Plant Stability Analysis by D. Buden and R. F. Miller (G. E. Co.); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 237–244; May, 1961.

A power plant with a heat exchanger such as a nuclear reactor substiuted for the conventional chemical interburners in a jet engine will cause a considerable change in dynamic performance. The instantaneous power generated by the heat source is not the same as the instantaneous power delivered to the turbine. The basic control problems are analyzed using fixed control parameters and partial derivatives around a given operating point. A mathematical criterion is developed and correlated with power plant test data. An understanding of the inherent limitations of combining a reactor, or any heat exchanger having a thermal lag, with a basic jet engine makes it possible to devise a means of control. The introduction of an effective operational speed control permits operation of a complete power plant under any desired condition.

1754
Distributed Parameter Network Synthesis Using the Potential Analog Computer by A. Acampora (Microwave Res. Inst.); *U. S. Govt. Res. Repts.*, vol. 35, p. 728(A); June 16, 1961. PB 150 906 (order from LC mi$3.30, ph$7.80.)

A transformation of the complex variable, termed a lambda transformation ($\lambda = \Sigma + j\Omega$), whereby networks consisting of lossless, commensurate transmission lines and resistors are directly expressible as rational functions of the transformed variable λ, is discussed, and a potential analog computer based on this transformation is described. The computer can be used to achieve networks that approximate a prescribed amplitude and phase response over a defined range of real frequency. The distributed parameter approximating structure is arbitrary in form and number of elements except that it consists of lossless, commensurate lines and resistors. The structure of the resulting networks is specifically prescribed at the outset of the approximation procedure. In the latter case, the particular networks considered are cascaded, lossless-line two-ports terminated in pure resistance.

1755
Computer Simulation of Aggregate Formation by M. J. Vold (University of Southern California); *U. S. Govt. Res. Repts.*, vol. 35, p. 554(A); May 16, 1961. PB 153 667 (order from LC mi$2.40, ph$3.30).

Computer generated model sediments of spherical particles, formed by addition of one particle at a time to the sediment, appeared in previous work to represent the behavior of physical model systems composed of microscopic glass beads adequately. However, sediment densities of semicolloidal silica in hydrocarbon solvents are too low by a factor of about 30 to be represened by such as model. Observation suggests that the model must take flocculation and sedimentation into account simultaneously. Generation of model flocs and exploration of their shapes, structures and densities are reported. Particle-by-particle addition to the floc was assumed. A sediment formed on the previous pattern from these flocs rather than from primary particles is still more dense than experimental sediments, but by a smaller factor (about 6). A more sophisticated model is under study.

1756
A Flow Pattern at High Subsonic Speeds Past a Wedge at Incidence in a Free Stream and a Choked Channel by J. B. Halliwell (Royal College Sci. Tech.); *J. Math. Phys.*, vol. 40, pp. 1–22; April, 1961.

The equations of flow are formulated in terms of dual integral equations, and the boundary value problem is formulated for a flow in an unchoked channel. The analyses for flow in a free stream or a choked channel follow as special cases of the general formulation. In the latter case, an iterative procedure is used to provide a full solution.

1757
The Prospects for the Utilization of Informational-Logical Machines in Chemistry (USSR) by L. I. Gutenmaker and G. E. Vleduts; *J. Assoc. Comp. Mach.*, vol. 8, pp. 240–251; April, 1961.

Russian investigations into the processing and retrieval of large volumes of information relating to chemistry are outlined. Input information is translated into a suitable form for machine storage. Inquiries are translated into machine language and placed in working storage before being processed. Algorithms for encoding chemical structure are being developed. With the aid of such algorithms, the searching for all compounds and synthesis procedures with specified properties will be facilitated. It is hoped to develop the system to combine calculation with information search to handle more complex problems.

1758
Computation of Complicated Combustion Equilibria on High-Speed Digital Computer by D. S. Villars (Naval Ord. Test Sta.); *U. S. Govt. Res. Repts.*, vol. 35, p. 767(A); June 16, 1961. PB 153 930 (order from LC mi$2.70, ph$4.80).

A reliable and rapid procedure of successive approximations which does not require any qualitative estimates of final answers as input data is described. Current values of concentrations are used to calculate all equilibrium constants. That reaction showing greatest fractional discrepancy between calculated and given equilibrium constants is selected by the program to be computed on the assumption of negligible interactions with the other equilibria. The process is repeated until the maximum discrepancy has been reduced to a value less than an error specified as a parameter of the problem. The program will accept any number of reactions and species. A matrix manipulation may be made to check that the chemical equations are in balance. The machine is then fed a list of chemical equations to be considered for a particular problem. It condenses the original matrix into a smaller working matrix consisting of rows and columns corresponding only to the actual elements and reactions and the actual chemical species to be involved in that particular problem. Speed of convergence is enhanced by transforming the chemical equations to reactions of formation of the various species from components existing at maximal concentrations at equilibrium. After checking that a proposed set of components is tenable, the program conducts the transformation to the new set of chemical equations and obtains the corresponding equilibrium constants. The program is also provided with the capability of computing enthalpies and entropies of the individual species for given temperatures and of the corresponding equilibrium mixture.

1759
Digital Circuit for Measurement of Spectral Line Intensities by S. Minami, H. Yoshinaga, and S. Fujita (Osaka University); *J. Opt. Soc. Am.*, vol. 51, pp. 674–678; June, 1961.

A digital circuit for measuring the integrated dc signal from a photosensitive device is described. The principal part is an analog-to-digital converter using a step integrator. With the auxiliary circuit added, the converter has nonlinear characteristics, which linearizes a nonlinear input. The long-term lack of reproducibility of the analog-to-digital conversion is less than ± 0.1 per cent of the full-scale count. The nonlinear analog-to-digital conversion system, for instance, makes possible a direct readout of the element concentration in the emission spectrochemical analysis.

1760
Mathematical Analysis and Digital Computer Solution of Natural Frequencies and Normal Modes of Vibration for a Compound Isolation Mounting System by L. Katz (David Taylor Model Basin); *U. S. Govt. Res. Repts.*, vol. 35, p. 735(A); June 16, 1961. PB 154 506 (order from LC mi$2.70, ph$4.80).

The mathematical analysis and solution of the natural frequencies and normal modes of vibration for a compound isolation mounting system by McGoldrick's method are discussed. The system consists of an assembly supported by a set of isolation mountings carried by a cradle which is, in turn, supported by another set of isolation mountings attached to the hull of a ship. The solution for a single, resiliently mounted, rigid body is presented as a special case of the two-body system.

1761
Recent Developments in Some Non-Self-Adjoint Problems of Mathematical Physics by C. L. Dolph (University of Michigan); *Bull. Amer. Math. Soc.*, vol. 67, pp. 1–69; January, 1961.

An extensive survey of recent developments in non-self-adjoint problems of mathematical physics is presented. Certain problems in quantum mechanics which, when segmented, involve non-self-adjoint operators are outlined and discussed. Problems in scattering theory, neutron diffusion, Sturm-Liouville systems, Livsic operators, dissipative operators, Wiener-Hopf integral equations, and Cauchy integral equations, are surveyed and discussed. Variational principles for treating non-self-adjoint problems are outlined.

1762
A Program for the Numerical Integration of the Boltzmann Transport Equation, NIOBE by S. Prieser, G. Rabinowitz, and E. deDufour (Nuclear Dev. Corp. of America); *U. S. Govt. Res. Repts.*, vol. 35, pp. 599–600(A); May 16, 1961. PB 152 587 (order from LC mi$3.90, ph$10.80).

A code (NIOBE) for numerically integrating the time-independent neutron or γ-ray Boltzmann transport equation which has been written for the IBM 704 is given. The code will calculate angular distributions, total fluxes, and currents for neutrons or photons as a function of energy or wavelength in a finite, multilayered, spherically symmetric configuration.

11. ANALOG AND HYBRID COMPUTERS

1763
An Accurate Analog Multiplier and Divider by E. Kettel and W. Schneider (Telefunken Res. Inst., Germany); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 269–272; June, 1961.

In the time-division multiplier, the product $x_1 \cdot x_2$ is formed by pulse-duration modulation with x_1 and amplitude modulation with x_2. The circuit can be arranged in such a manner that division by means of a quantity x_3 can be carried out simultaneously, the output being $x_1 \cdot x_2/x_3$. When transistor switches are employed, the error is only $1 \cdot 10^{-4}$ machine units. The zero error for x_2, used for amplitude modulation, can be reduced to $2 \cdot 10^{-5}$ machine units.

1764
Application of the Magnetoresistance Effect to Analog Multiplication by J. M. Hunt (Stanford Electronics Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 602(A); May 16, 1961. PB. 149 333 (order from LC mi$6.90, ph$21.30)

Bounds on the performance of a magnetoresistance analog multiplier are studied. Emphasis was placed upon the following areas: 1) analysis of multiplier error in terms of constituent errors, 2) theoretical analysis of the source of certain errors caused by imperfections in the paired magnetoresistor unit which is the novel element of the multiplier, and 3) fabrication and testing of sufficient magnetoresistor elements to permit the establishment of a realistic bound on the error component contributed by these elements. Attention was also devoted to methods for minimizing multiplier errors without recourse to external corrective circuitry.

1765
Automatic Servo-Driven Bridge Measures Analog Voltage Ratios by S. Shenfeld and H. R. Manke (U.S. Naval Underwater Sound Lab.); *Electronics*, vol. 34, pp. 53–55; July 28, 1961.

An automatic ratio bridge which measures the ratio of analog voltages, accurate to 0.002 per cent, in a 400-cycle analog computer is described. The main features of the bridge include a rotary auto-transformer with direct indication, a servo system for automatic nulling and maximum slewing time of 30 sec (full range coverage). The major design problem is the removal of the quadrature component of voltages which tend to mask the true null adjustment. This is accomplished by adding a transistor chopper with a low cutoff frequency filter network.

Optimization Problems: Solution by an Analogue Computer—*see* 1739.

1766
A Method for the Research of the Zeros of a Polynomial with an Analog Computer by A. Lepschy (Trieste University); *Alta Frequenza*, vol. 30, pp. 216–218; March, 1961.

A method for determining the zeros of a polynomial by means of an analog computer is presented. The method is based on harmonic synthesis of the polynomial. Sinus-

oidal voltages corresponding to the variable and to its powers are obtained as outputs of cascade operational amplifiers and are combined with the output of another operational amplifier to give a sinusoidal voltage corresponding to the polynomial. The transfer function of each operational element may be changed by varying the position of the input potentiometer of the operational amplifier and the frequency of the feeding voltage. The roots of the polynomial correspond to the values of the transfer function for which the output of the operational combiner is zero. The main characteristics of the method are described.

1767
Analog Computers' Role in the Processing Industries; *Automatic Control*, vol. 14, pp. 45–47; June, 1961.

Applications of analog computers and components in the processing industries are outlined. The devices can be used to perform general computation (*e.g.*, solving simultaneous differential equations), to simplify measurement techniques, and to function as units in complex data-handling and digital control systems. The machines are also useful in industrial process simulation and control system analysis. Another application concerns the saving of effort in design calculations, especially those involving trial-and-error procedures.

Shaft-Angle Digitizing and Recording System—*see* 1793.

1768
High-Speed Analog-to-Digital Converters Utilizing Tunnel Diodes by R. A. Kaenel (Bell Labs.); IRE TRANS. ON ELECTRONIC COMPUTERS, vol. EC-10, pp. 273–284; June, 1961.

Two analog-to-digital sequential converters which combine in one tunnel-diode pair per bit the functions of an amplitude discriminator and memory are described. In addition, one of the two schemes utilizes each tunnel-diode pair as a delay network. The conversion duration of one of these 6-bit converters, which employs germanium 2N559 transistors and gallium arsenide 1N651 tunnel diodes, has been set to 1 μsec. Shorter conversion times are possible. Particular emphasis is given to the discriminator property of series-aiding tunnel-diode pairs. A comprehensive bibliography relating to tunnel-diode switching circuits is attached.

Digital Circuit for Measurement of Spectral Line Intensities—*see* 1759.

12. REAL-TIME SYSTEMS AND AUTOMATIC CONTROL; INDUSTRIAL APPLICATIONS

Terminal Control System Applications—*see* 1794.

1769
A General Performance Index for Analytical Design of Control Systems by Z. N. Rekasius (Purdue University); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 217–222; May, 1961.

A performance index which enables one to specify the desired response of the optimum system in terms of the differential equation describing the response of an ideal model is proposed, and a simple straightforward procedure of calculating this performance is outlined. This procedure consists of solution of a set of linear algebraic equations. Simultaneous solution of these algebraic equations yields the value of the performance index in terms of gain and time constants of the actual system. It is then a simple matter to calculate the numerical values of the free gain and time constant parameters for the optimum system (*i.e.*, to minimize the performance index). The procedure of optimization is illustrated by means of an example of a third-order system.

1770
On a Property of Optimal Controllers with Boundedness Constraints by H. L. Groginsky (Raytheon Co.); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 98–110; May, 1961.

A theory for optimal control systems designed to operate a plant of known characteristics is described. It is assumed that only limited changes in the system characteristics can be effected by the control variables at the designer's disposal. It is shown that within the assumed limitations for a wide class of inputs and systems and for a certain class of measures of the system performance, an optimal system behaves as a relay or switched system during the transient period and as a continuous system during periods in which the input is reproduced identically. A procedure for determining the switching times during the transient period in terms of the permissible measurements is described.

1771
Theory and Design of High-Order Bang-Bang Control Systems by M. Athanassiades and O. J. M. Smith (University of California); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 125–134; May, 1961.

A complete analysis and design of a nonlinear controller to minimize the response time of a limited input plant whose transfer function has N real roots is presented. The design procedure is based exclusively on the concept of the switching hypersurface of the system in N-dimensional state space and on the concept of the "distance function" from the state point to the switching hypersurface. The linear and nonlinear transformations performed by a nonlinear computer upon the error and its time derivatives in order to generate the optimal amplitude limited input to the plant, are described in detail, and properties of the switching surfaces, which are subsets of the switching hypersurface, are also described.

1772
Model Feedback Applied to Flexible Booster Control by G. E. Tutt and W. K. Waymeyer (Douglas Aircraft); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 135–142; May, 1961.

A feedback model approach to the design of flexible space vehicle boosters involving the control of an aerodynamically unstable airframe in a dynamic wind shear (jet stream) environment is described. This system does not adapt to body bending, but instead is contrived to ignore it. A conventional attitude control system is assumed in which rigid body control considerations have been used to design the control loops. A model of the plant (airframe rigid body dynamics) is derived. The attitude and rate feedbacks are now synthesized by a combination of actual and model rate and attitude. Bending and similar dynamic effects are filtered from the actual attitude and rate signals as required, and the information thus rejected is supplied from the model of the plant. It is readily shown that the performance of this system in response to commands is substantially identical in all important respects to the original rigid body system.

1773
Transfer-Function Tracking and Adaptive Control Systems by C. N. Weygandt (University of Pennsylvania) and N. N. Puri (Drexel Inst. Tech.); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 162–166; May, 1961.

In an adaptive system in which plant parameters are varying, it is necessary to track or measure the plant parameters. Two separate schemes are proposed for tracking the transfer function of a multiorder system. The first scheme is based on perturbing the normal process with a small-amplitude sinusoidal perturbation signal consisting of different frequencies. The tracking system is a closed-loop system. The second scheme does not depend upon a perturbation signal, but does require knowledge of the form of the transfer function of the system. It is more suitable for tracking systems which have a number of first-order log units connected in tandem.

1774
Precision of Impulse-Response Identification Based on Short, Normal Operating Records by R. B. Kerr and W. H. Surber, Jr. (Princeton University); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp.173–182; May, 1961.

An identification scheme for estimating in a short time the impulse response of a system from normal operating records is presented. Maximum-likelihood estimates of the impulse response are discussed, and a "sufficiency" criterion on the input signal is defined based upon the expected integrated squared difference between the actual and estimated impulse responses. Examples of sufficient and insufficient test signals are given in terms of a "sufficient record length" criterion. Experimental results illustrating this latter criterion are presented. The time variation of the system parameters sets an upper bound on the useful record length. Some preliminary results relating this time variation to the useful record length are also presented.

1775
A Technique of Linear System Identification Using Correlating Filters by W. W. Lichtenberger (University of Illinois); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 183–199; May, 1961.

Random signals and cross correlation can be used to determine the impulse response of a system. If, instead of a random testing signal, one is employed which has certain properties similar to the random signal, a linear filter may be constructed which performs the necessary cross correlation. No multiplication is involved, and the output is a continuous function of time. Thus, a signal filter obtains the impulse response for all

values of time. A disadvantage of this method is that in a practical situation, there is usually a great amount of noise present. This "noise" consists mostly of process actuating signals since the measurements must be made "on line."

1776
A Minimal Time Discrete System by C. A. Desoer and J. Wing (University of California), IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 111–125; May, 1961.

A sampled-data control system with a plant having real poles and limited input is considered. The plant forcing function which will bring the system to equilibrium in a minimum number of sampling periods is desired; this function is called an optimal strategy. To implement this particular optimal strategy, a surface in state space called the critical surface is defined. It is shown that this optimal strategy will be generated by the following procedure: at the beginning of each sampling period, the distance ϕ from the state of the system to the critical surface is measured along a fixed specified direction; if $\phi \geq 1$ (or ≤ -1), then the forcing function for that sampling period is $+1$ (or -1); if $|\phi| < 1$, then the forcing function is ϕ. For a third-order plant, it is shown that the critical surface has certain properties which lead to a simple analog computer simulation.

1777
A Parameter-Perturbation Adaptive Control System by R. J. McGrath (Aerospace), V. Rajaraman, and V. C. Rideout (University of Wisconsin); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 154–162; May, 1961.

Theoretical and simulator studies of some new forms of a parameter perturbation self-adaptive system are presented. Here, the response of the control system is subtracted from that of a reference model to obtain the error signal. As in the previous studies, the controllable parameters of the system are sinusoidally perturbed. Somewhat different methods of processing the perturbation and error signals have been employed to obtain the parameter control signal which is used in the automatic optimization of the control system. Computer studies are made with both random and deterministic input signals and parameter disturbances. The results are compared with the analytical results.

1778
Adaptive Servo Tracking by A. I. Talkin (Diamond Ord. Fuze Labs.); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 167–172; May, 1961.

A self-adaptive sampled-data radar tracking loop is described. The tracking loop may be considered to be a low-pass filter with a variable bandwidth. The loop is designed to adapt rapidly to changes in the input signal by monitoring both the apparent error and the loop output. Results show a mean tracking error 25–34 per cent less than that of a comparable linear system, at a receiver SNR of 10 db.

1779
Stability of Servomechanisms with Friction and Stiction in the Output Element by P. K. Bohacek and F. B. Tuteur (Yale University); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 222–227; May, 1961.

Servomechanisms with friction in the output element are often observed to oscillate, even though the Bode diagram indicates stability. The conditions for this instability and the type of oscillation that can occur are investigated. It is found that an overdamped system with a lag equalizer is stable if $L < 2C/C-1$, where L is the lag ratio and C = static friction + Coulomb friction; with a lag-lead equalizer, it is stable if

$$\frac{L}{1+a/b} < \frac{2C}{C-1'}$$

where a/b is the ratio of the two zeros of the network. Experimental results that correlate with the theory are also included.

1780
A Modified Lyapunov Method for Nonlinear Stability Analysis by D. R. Ingwerson (Sperry Rand Co.); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 199–210; May, 1961.

The Lyapunov stability criterion deals with arbitrarily small disturbances. A generalization of the original theorem which applies to arbitrarily large and small disturbances, and to intermediate conditions as well, is given. In contrast to the success that has been achieved in advancing the theoretical concepts of stability by this method, little has been accomplished in the way of formulating practical means for applying them to specific problems. A method which is easily applied to many of the systems encountered in automatic control and which has given good results for numerous examples is presented.

1781
Sampled-Data Deviations in a Vertical Indicator with a Stellar Monitor by T. M. Pienkowski (Wright-Patterson AFB); *U. S. Govt. Res. Repts.*, vol. 35, p. 718(A); June 16, 1961. PB 171 609 (order from OTS $3.00).

The use of a digital computer or a star tracker in a vertical indicator results in a sampled-data system in contrast to the usual continuous data system, thus introducing errors. These errors are explained and computed by a linear analysis and presented in accurate graphical form for use as a design aid. The error resulting from nonlinearities in the system is also considered.

1782
Sensitivity Considerations for Time-Varying Sampled-Data Feedback Systems by J. B. Cruz, Jr. (University of Illinois); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 228–236; May, 1961.

A synthesis procedure for linear time-varying sampled-data feedback systems is described. The plant and compensators are characterized by transmission matrices first introduced by Friedland. Part of the specification involves a deviation or error matrix for the time-varying plant and an allowable deviation matrix for the closed-loop system. Noise considerations are also included. Using a technique analogous to that of Horowitz which was originally used for multivariable continuous systems, the digital compensator transmission matrices are derived. The corresponding time-varying digital compensators are realized by means of zero-order hold circuits, switches, resistors, and adders. An example using a digital computer simulation is included.

1783
Low-Speed Time-Multiplexing with Magnetic Latching Relays by J. F. Meyer (Calif. Inst. Tech.); IRE TRANS. ON SPACE ELECTRONICS AND TELEMETRY, vol. SET-7, pp. 34–41; June, 1961.

Many satellite and spacecraft telemetry systems require that measurements be time-shared at relatively low rates (less than one sample per second) and that several rates be available in order to minimize redundant sampling. A multiplexing system designed specifically for low-speed operation and multiple-rate flexibility so as to gain advantage in other critical areas of performance is discussed. A unique synthesis of the basic circuit provides for the time-multiplexing of n measurements with $n-1$ magnetic latching relays. Assuming an external two-phase clocking source, no other components, active or passive, are required in the circuit. Other advantages of the circuit are: 1) low average power consumption; 2) no additional monitoring or reset circuitry required to insure proper operation at turn-on or after momentary power failure; and 3) the virtual impossibility of switching more than a single measurement to the common output line, even in the case of component or wiring failure.

1784
High Speed Data System Solves Low Level Signal Problems by W. F. Kamsler (Epsco-West); *Automatic Control*, vol. 14, pp. 37–44; June, 1961.

A high-speed digital data acquisition system capable of handling 200 channels of input from three types of sources: strain gauge bridges, thermocouples, and miscellaneous millivolt signals, is described. Input data are recorded digitally on magnetic tape at a rate of 5000 or 10,000 data samples per second, and then can either be transcribed into computer format which is acceptable to an IBM 704 or 709 computer, or can be presented on analog plotters. Detailed system descriptions are included.

1785
Phase Reversal Data Transmission System for Switched and Private Telephone Line Applications by E. Hopner (IBM Corp.); *IBM J. Res. & Dev.*, vol. 5, pp. 93–105; April, 1961.

A phase reversal data transmission system capable of operating at 2000 bauds over private telephone lines and at 1200 bauds over switched networks is described. The advantages and limitations of the system are discussed from a theoretical and practical standpoint. The clocking problem in data transmission is considered and several approaches are indicated. Extensive field tests over switched and private lines in Europe and on private (SAGE) lines in the U. S. are summarized.

1786
Automatic Digital-Data-Error Recorder by E. J. Hofmann (Lincoln Lab.); IRE TRANS. ON INSTRUMENTATION, vol. I–10, pp. 27–31; June, 1961.

An automatic digital-data-error recorder (ADDER) which automatically detects and records errors occurring during the transmission of digital information over data circuits is described. Information of known structure is transmitted at one end of a data channel and compared at the output by means of the ADDER. The ADDER is composed of analog-to-digital converters, shift registers, counters for storing information, comparator circuits for error detection, and sequencing logic to control the shifting, storing, and punching out of information. Flip-flop storage is used to store a maximum of 126 bits of information. The following information in regard to the transmission performance of the over-all system is obtained: 1) the number of words in error and their time distribution, 2) the number of bits in error in erroneous words and their position, and 3) the relative occurrence of lost or gained sync (or start) pulses. The output of the device is punched paper tape which, through the use of a suitable program, may be analyzed by an IBM 709 or a similar computer.

1787
Carrier Phase Reversal Transmits Digital Data over Telephone Lines by J. R. Masek (Hallicrafters); *Electronics*, vol. 34, pp. 56–58; May 26, 1961.

A transmission system which handles digital data, together with timing and synchronization signals, in a single channel is described. Digital data are put into groups of 4 bits, and then coded into code words of 5 bits for each 4 bits group. A special code word (10000) is used as the synchronization signal. The code words are so selected that no more than 3 zeros or 7 ones can occur in succession in the coded data signal; therefore, timing pulses can easily be regenerated from one/zero crossovers. A 1950-cps carrier signal is modulated by the digital data such that a one is represented by a reversal of carrier phase and a zero is represented by no phase reversal. At the receiving end, the transmitted data are recovered from the carrier by synchronous detection.

1788
Present Address Selection for DIGICOM, a Digital Communication System by R. Frank (USASRDL); *U. S. Govt. Res. Repts.*, vol. 36, p. 21(A); July 5, 1961. AD 255 267 (order from OTS $3.60).

1) A method for decreasing the amount of time a telephone caller may have to wait for service is discussed. 2) The design of the necessary circuitry to adapt DIGICOM for operation with preset address selection is described. It is concluded that preset address selection has many advantages and that a completely new DIGICOM should be built to show the full capabilities of this method.

Analog Computer's Role in the Processing Industries—*see* 1767.

1789
Cold-Cathode Tube Circuits for Automation (Part 3) by R. S. Sidorowicz (Hivac Ltd); *Electronic Engrg.*, vol. 33, pp. 296–302; May, 1961.

Schematics of oscillator circuits, multivibrator circuits, a free-running square-wave generator circuit, and a numerical indicator drive circuit employing cold-cathode tubes are presented. Some examples of cold-cathode tube systems are described, including an electronic dial which is capable of generating a preset number of pulses, a controlled counter which will count the number of batches of objects, and an electronic inspection unit in which an object is inspected by photosensors and is accepted or rejected according to an established criterion.

13. GOVERNMENT, MILITARY, AND TRANSPORTATION APPLICATIONS

1790
Digital Computer Program for Superelevation Cam Design by M. Archambault (Dynamic Simulations Lab.); *U. S. Govt. Res. Repts.*, vol. 35, p. 738(A); June 16, 1961. PB 153 866 (order from LC mi$4.80, ph$13.80).

A digital computer program for the rapid calculation of manufacturing data essential to the design of preproduction cams which are utilized in ballistic computers of tank fire control systems is presented. The cam profile generated introduces the superelevation angle required by the tank main armament for a particular type of ammunition.

1791
Combat Vehicle Firing Stability (Active Suspension) by C. M. Fischer (U. S. Army); *Commun. Assoc. Comp. Mach.*, vol. 4, pp. 279–283; June, 1961.

A mathematical model for combat vehicle response during the firing cycle of main combat weapons is proposed. Two coordinate systems are used, one fixed in space and a second relative to the vehicle itself. The equations of motion are set up relative to the moving axes, and the appropriate transformations are subsequently applied. The problem reduces to the solution of a set of six second-order differential equations, which are solved with the aid of a modified fourth-order Runge-Kutta process.

1792
Three Dimensional Flight Table Device No. 24-X-5 by M. Kanes (Bendix Aviation Corp.); *U. S. Govt. Res. Repts.*, vol. 35, p. 601(A); May 16, 1961. PB 154 806 (order from LC mi$1.80, ph$1.80).

A simulator for use with a master computer for the purpose of simulating the flight of fast air-to-air missiles on a 1-to-1 time scale is described. The unit consists of a high-performance, servo-controlled gimbal system which follows command inputs from an analog computer. Missile sections and components mounted on these gimbals are subjected to the torques, angular accelerations, and angular velocities which would be experienced in the flight being simulated. Instruments on the gimbal system monitor velocity and position, and compute Euler angle transformations.

1793
Shaft-Angle Digitizing and Recording System by E. Nadalin (Naval Ord. Test Station); *U. S. Govt. Res. Repts.*, vol. 35, p. 739(A); June 16, 1961. PB 153 647 (order from LC mi$3.90, ph$10.80).

A system for recording simultaneously in digital form the range, azimuth, and elevation parameters of target position in relation to radar position, together with range timing on magnetic tape in an IBM-709 tape format is reported The parameters are digitized by means of shaft-angle encoders. These data are stored in a magnetic-core memory by means of transistorized logic circuitry and then shifted onto magnetic tape. The data are now in a format for direct entry into the IBM 709 computer.

A Mean-Weighted Square-Error Criterion for Optimum Filtering of Nonstationary Random Processes—*see* 1740.

1794
Terminal Control System Applications by E. A. O'Hern and R. K. Smyth (Autonetics); IRE TRANS. ON AUTOMATIC CONTROL, vol. AC-6, pp. 142–153; May, 1961.

Certain theoretical extensions of earlier work on terminal control techniques and their application to an aircraft landing system are described. The case considered is of a system having a second-order response from altitude rate command to altitude rate. The terminal time equations for this case are developed together with the various closed-loop weighting functions by transform methods. From the terminal equations developed, the terminal controller equations are synthesized for a two-condition terminal controller which controls altitude and altitude rate at the terminal (touchdown) time. The mechanization of these terminal controller equations is presented. The flight test results of the terminal control system using this system are described briefly. A comparison between flight test and simulation results is included.

1795
Air-Traffic Terminal Controller Computes and Sends Instructions by R. Meuleman and S. D. Moxley, Jr. (Avco Corp.); *Electronics*, vol. 34, pp. 47–51; May 26, 1961.

An air-traffic terminal control system that simultaneously controls 18 arriving and 6 departing aircraft in a terminal area is described. An individual tracking-computing channel is assigned to each arriving aircraft. The information about the position of the aircraft is tracked by radar and is supplied to the flight-path electromechanical analog computer. The computer schedules the time for the aircraft to land and directs the aircraft to the right course. The system can talk directly to the pilot so that the human controller's work-load is significantly reduced. Prerecorded voice phrases are stored on a magnetic drum. These phrases are selected, combined into messages, and transmitted by an automatic voice unit having a vocabulary of 25 different message types with almost unlimited numerical combinations.

14. BUSINESS APPLICATIONS

1796
The Use of Pegasus Autocode in Some Experimental Business Applications of Computers by H. W. Gearing (Metal Box Ltd.); *Computer J.*, vol. 4, pp. 30–34; April, 1961.

The experimental application of the Pegasus autocode to such commercial problems as market surveys, experimental sales forecasts and quality control statistics is described. The relatively short time used for program development and result tabulation demonstrates the potential advantages of the sophisticated use of computers in business environments.

Abstracts of Current Computer Literature

THREE-YEAR CUMULATIVE SUBJECT AND AUTHOR INDEX

THREE-YEAR CUMULATIVE AUTHOR INDEX

Three-Year Cumulative Subject Index

J

L

Three-Year Cumulative Author Index

D

E

F

G

Q

R

S

PGEC News

To All PGEC Chapter Officers and Committee Chairmen

The PGEC *News* section of the TRANSACTIONS is open for your announcements and reports of activities. Deadline is the first of the month, two months ahead of the date of issue. Send all items to the *Editor*.

THE CHAIRMAN'S LETTER

St. Paul, Minn., November 10, 1961—

AFIPS and the Role of PGEC

You will be reading this shortly after the 1961 Eastern Joint Computer Conference, which was the first national computer conference sponsored by the American Federation of Information Processing Societies. The debut of AFIPS is a significant occasion for the information processing community, and we wish the new organization well in carrying out its important mission. Now that AFIPS is beginning to function, it is appropriate to scrutinize the role of PGEC. The PGEC AdCom has been doing a great deal of this during recent months, and gave the subject a pretty thorough workout at Los Angeles in September.

Here is what we see in our mirror. In publications the feeling is that our TRANSACTIONS is a pretty high-grade journal, and that our editors have been doing a tremendous job. What about conferences? PGEC participation in the JCC's is strong, but our role is somewhat obscured from public view, and this is especially true with the debut of AFIPS. At the IRE International Convention and at WESCON, even the presence of several well-organized computer sessions does not draw a high concentration of computer people. In the several national symposia which PGEC cosponsors, both our initiative and our degree of participation should be far stronger. What about Chapter activities? Many of the Chapters are seriously in need of purpose and motivation.

PGEC Symposia

As a positive step, we discussed the desirability of one or more regular symposia sponsored exclusively by PGEC, and organized and operated at the Chapter level. The need was expressed for a type of conference which is specifically geared to the interest of PGEC members. Sponsorship by a Chapter is natural and practical, since close liaison among the arrangers is highly desirable. Keith Uncapher suggested that the Orange County (Calif.) Chapter of PGEC is ideally situated to set the example by launching a good symposium. This Chapter, although relatively new, is blessed with members and officers who are long on conference and technical experience.

A few weeks later, Keith and I had the opportunity to talk over the proposition with Bill Gunning and his Orange County colleagues. These fireballs immediately took over; at this writing, plans for a symposium are solid. The symposium topic is "Engineering Techniques in Missile and Spaceborne Computers." Bill Gunning is conference chairman. Date and location are not final at this writing; tentatively, late October, 1962, in Anaheim, Calif. National participation is intended and encouraged. The program will consist primarily of invited papers. A great deal of enthusiasm has already been generated. More about this next time.

Technical Activities Committees

A major item for the December meeting of the PGEC is to consider and adopt a plan for establishing Technical Activities Committees in special-interest topical areas which lie within the scope of PGEC. According to the thinking at this writing, each Technical Activities Committee (TAC) will provide the leadership in organizing technnical activities in its specialty area, including:

Sponsoring or cosponsoring national symposia, using PGEC Chapters as local arrangements committees where appropriate;

Handling PGEC participation in international meetings relating to the specialty;

Organizing and soliciting papers for specialized sessions at IRE Conventions, JCC's, etc.;

Soliciting and reviewing TRANSACTIONS papers in the specialty;

Sponsoring special issues of the TRANSACTIONS;

Cooperating with IRE technical committees (like Committee 8) in establishing IRE Standards.

We hope to keep the rules and procedures under which the TAC's operate as simple as possible. To insure that each TAC is manned with able, motivated workers, a new TAC will be activated only on petition of some reasonable minimum number of qualified workers in the specialty. A TAC is not a list of all PGEC members casually interested in a given specialty; it is, rather, a vigorous steering committee for organizing technical activities. For guidance purposes, there will be an initial list of specialties in which TAC's would be desirable; however, it is by no means necessary to adhere to the categories in this list. The plan which we adopt must serve the needs of the membership and provide for the natural growth of specialties within specialties.

Does the Name Fit the Scope?

During the course of our self-scrutiny, the suggestion came up that it would be appropriate to rename PGEC to PGIPS (Information Processing Systems). If you look over the PGEC scope which we adopted last March (see the June TRANSACTIONS, p. 337), would you feel that the name PGIPS is more representative of the technical area we are serving? The proposed title picks up "information processing," which has a broader connotation than "computers," and adds the important world "systems." The combination gives recognition to the long established fact that our attention is not confined to components and techniques internal to computers. Further, it is certainly our responsibility, whether as PGEC or as PGIPS, to serve the increasing interest in system engineering of computer-centered systems.

PGEC Annual Meeting

The next meeting of the AdCom will be the Annual Meeting of PGEC, at New York, N.Y., during the 1962 IRE International Convention.

ARNOLD A. COHEN
Chairman, PGEC

CHAPTER NEWS

The two neighboring chapters in the Los Angeles vicinity, the Los Angeles Chapter and the Orange County Chapter, announce the following officers for 1961–1962:

Los Angeles Chapter

Chairman—Paul M. Davies, Abacus, Inc.
Vice Chairman—Dr. Robert Kudlich, AC Spark Plug.
Secretary-Treasurer—Dr. Ike Nehamma, Rand Corp.
Program Chairman—Viv Nininger, Librascope Corp.
Membership Chairman—Dave Hartig, Librascope Corp.
Arrangements and Publicity—Ronald Leuschner, Hughes Aircraft Co.
Student Relations—Donald Segel, Marquardt Co.
Post Chairman—Mal Davis, Rand Corp.

Orange County Chapter

Chairman—William Gunning, Astrodata Inc.
Vice Chairman—Charles Hobbs, Aeronutronics.

IFIPS NEWS

INTERNATIONAL MULTILINGUAL COMPUTER GLOSSARY

Through the collaboration of two international organizations, a "Multilingual Glossary of Automatic Data Processing Terminology" is being developed which will have international significance. The two organizations responsible for preparing the glossary, the International Federation of Information Processing Societies and the Provisional International Computation Centre, have agreed to make the glossary available, upon completion, to the International Organization for Standardization.

At its May meeting in Geneva, the newly-formed ISO Technical Committee No. 97 on Digital Computers and Data Processing Machines formally requested that IFIPS and PICC assume responsibility, on its behalf, for the preparation of this multilingual glossary. The ISO Technical Committee agreed to accept the glossary as a first draft of an International Standard Glossary.

The IFIPS Committee for the Standardization of Terminology and Symbols, under the chairmanship of G. C. Tootill of the Royal Aircraft Establishment, England, is undertaking two tasks in the preparation of the glossary. First, the IFIPS Committee is reviewing the latest draft of the British Standard's Glossary to determine what alterations to the concepts are necessary to make the glossary suitable for international use. Secondly, the committee is studying the glossary to insure its completeness and accuracy. Based on the work of the British Standards Institution, the Provisional International Computation Centre is preparing a draft of the multilingual glossary in five languages. Dr. J. E. Holmstrom, previously associated with UNESCO, is directing the project of the PICC.

It is hoped that the multilingual glossary, which will mark an important milestone in standardization of computer terminology, will be available by August, 1962.

Notices

This *Notices* Section is open to all who have an announcement of a conference, symposium, session, publication, or other artifact of interest to the PGEC membership. Please send announcements to the *Editor*, who will put them in the first available issue. The right is reserved to edit the announcements, and to decide whether they indeed are aimed at our audience.

Summary Calendar of Coming Events

January 28–February 2, 1962—AIEE Winter General Meeting, Statler Hotel, New York, N. Y.
February 6–7, 1962—ONR Symposium on Redundancy Techniques for Computing Systems, Washington, D. C.
February 14–16, 1962—International Solid-State Circuits Conference, University of Pennsylvania and Sheraton Hotel, Philadelphia, Pa.
March 26–29, 1962—IRE International Convention, Coliseum and Waldorf-Astoria Hotel, New York, N. Y.
April 24–26, 1962—PIB Symposium on Mathematical Theory of Automata, New York, N. Y.
May 1–3, 1962—Spring Joint Computer Conference, San Francisco, Calif.
May 3–4, 1962—International Congress on Human Factors in Electronics, Lafayette Hotel, Long Beach, Calif.
August 21–24, 1962—WESCON (Western Electronics Show and Conference), Los Angeles, Calif.
August 27–September 1, 1962—Second International Conference on Information Processing, Munich, Germany.
September 3–7, 1962—International Symposium on Information Theory, Brussels, Belgium.
December 4–7, 1962—Fall Joint Computer Conference, Bellevue Stratford Hotel, Philadelphia, Pa.
Items in boldface print are sponsored or cosponsored by PGEC.

FURTHER INFORMATION ON COMING EVENTS

AIEE WINTER MEETING

The AIEE Computer Systems Subcommittee will sponsor sessions on Kilomegacycle Computing Systems.

SYMPOSIUM ON REDUNDANCY TECHNIQUES FOR COMPUTING SYSTEMS

This symposium, sponsored by the Information Systems Branch, Office of Naval Research, will be held in the Department of the Interior Auditorium, C St. between 18th and 19th Sts., N.W., Washington, D. C.

The objective of the Symposium is to focus attention toward new ideas, research and developments which may lead to the introduction of redundancy techniques into forthcoming computing systems. The program will consist of papers invited from many of the organizations engaged in appropriate research and development activities.

Attendance is open to all interested technical personnel. Further information and a preliminary Symposium program, when available, may be obtained by contacting

Miss Josephine Leno
Code 430A
Office of Naval Research
Washington 25, D. C.
(Phone OXford 6-6213).

IRE INTERNATIONAL CONVENTION

Sessions sponsored by the PGEC will be held on March 26 and 27, 1962. Exhibits and technical sessions at the four-day meeting will cover the entire range of interests of the IRE.

PIB SYMPOSIUM ON MATHEMATICAL THEORY OF AUTOMATA

The Polytechnic Institute of Brooklyn has announced a Symposium on the Mathematical Theory of Automata. Professors A. E. Laemmel and E. J. Smith are cochairmen. The PGEC is one of the cosponsoring organizations.

SPRING JOINT COMPUTER CONFERENCE

The following officers have been announced:

General Chairman—George Barnard, Philco Western Development Laboratories.
Vice Chairman—Dr. Hewitt Crane, Stanford Research Institute.
Secretary-Treasurer—Robert Isaacs, Philco Western Development Laboratories.

The Chairman of the Technical Program Committee is Dr. Richard I. Tanaka Lockheed Missiles and Space Co. Vice-Chairman is Dr. Robert C. Minnick, Stanford Research Institute. John E. Sherman, Lockheed Missiles and Space Co., and R. J. Andrews, IBM, San Jose, Calif., are serving as associate chairmen for special sessions.

Both the technical program and the exhibits will be at the Fairmont Hotel, San Francisco, Calif.

INTERNATIONAL CONGRESS ON HUMAN FACTORS IN ELECTRONICS

The Congress, sponsored by the IRE PGHFE, has as its theme "Man-Machine Engineering: Methods, Models and Measurements." Technical discipline experts have been invited to present papers describing applications of their disciplines to human behavioral descriptions. Papers on new research findings will also be presented. The papers deadline has passed, but further information about the meeting may be obtained from the *Chairman:*

Dr. Charles Owen Hopkins
Hughes Aircraft Co.
Culver City, Calif.

IFIP CONGRESS 1962

Preliminary selection of American papers for consideration by the international program committee has been made. Nearly 400 abstracts were submitted by prospective authors. These covered a wide range of topics of interest to engineers, mathematicians, linguists, programmers, and businessmen.

At the same time, arrangements are being made for strong U. S. attendance. Chairman for U. S. Travel Arrangements is Dr. Werner Buchholz, IBM Corp., South Road Laboratory, Poughkeepsie, N. Y. Those who have not returned the travel questionnaire which was included in the mailing of the EJCC preliminary program are urged to communicate their travel needs to Dr. Buchholz so that his committee may make suitable plans.

INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY

An international Symposium on Information Theory, sponsored by PGIT, and organized by the Benelux Section of IRE, and SITEL is to be held in Brussels, Belgium, September 3 to 7, 1962.

The following is a tentative list of subjects:

Coding and decoding of digital and analog communication.
Studies of random interferences and of information-bearing signals.
Compression.
Analysis and design of communication and detection systems.
Pattern recognition, learning, adaptive filters.
Automata and other forms of information processing systems.
Processing of sensory information. Human operators.
Nervous systems.
Linguistics.
Scientific method.

It is hoped that all papers can be printed before the Symposium. The following deadline schedule is necessary:

Receipt of 500–1000 word abstracts: January 15, 1962.
Receipt of full-length papers: April 15, 1962.

Authors will be notified of the preliminary acceptance of their abstracts by February 1, 1962.

Abstracts and papers should be submitted to the Chairman of the Organizing Committee, Dr. F. L. Stumpers, Phillips Research Laboratories, Eindhoven, The Netherlands

PUBLICATIONS AVAILABLE

"International Repertory of Computation Laboratories," Published by Provisional International Computation Centre (PICC), Palazzo degli Uffici, Zona dell 'EUR, Rome, Italy

This International Repertory, which has, in the past, been published as part of the PICC Bulletin, has now been issued as a separate bound loose-leaf volume, priced at $6.50.

It contains detailed and up-to-date information—name, address and officers of each institution, type of equipment installed, as well as that contemplated, field of experience, training available, and periodical publications—on approximately 300 computing laboratories in 30 countries.

All entry sheets (one for each laboratory) are grouped by country, the countries classified in alphabetical order. There is also an alphabetical index by name of laboratory for easy reference.

Supplementary sheets will be supplied free of charge with subsequent issues of the PICC Bulletin.

COMPUTER PROJECT IN NORWAY

The Institute of Radio Techniques at Norway's Institute of Technology in Trondheim is undertaking the design and construction of a digital computer. The project will extend over a three-year period, starting about January 1, 1962. The team which is being assembled for the project hopes to acquire the services of an experienced computer engineer and would welcome applications. Available financial support includes post-doctorate fellowships for qualified candidates. Further details may be obtained from

Mr. Arne Lyse
Institute of Radio Techniques
Norway's Institute of Technology
Trondheim, Norway.

URGE YOUR FRIENDS TO JOIN THE PGEC

Advantages of Membership

1) Members regularly receive the IRE Transactions on Electronic Computers, which include the best technical papers on computers. The Transactions will be published bimonthly starting in 1962.
2) Members may participate in activities of local chapters of the PGEC.
3) Through the Transactions and the local chapter activities, members may keep up with the most recent advances in computer technology.

Who May Join

1) Those who belong to the IRE.
2) Those who belong to one of the following societies may become Affiliates of the PGEC:

American Management Society
American Mathematical Society
American Physical Society
American Society of Mechanical Engineers
Association for Computing Machinery
Institute of the Aeronautical Sciences
National Association of Accountants
National Machine Accountants Association
Operations Research Society of America
Society for Industrial and Applied Mathematics
Society of Automotive Engineers
American Institute of Electrical Engineers
Institution of Electrical Engineers (London)
Mathematical Association of America
Instrument Society of America.

How to Join

1) a) IRE members—make out a check for $4.00 payable to the IRE. This covers the PGEC fee.
   b) Non-IRE members desiring affiliate status—make out a check for $8.50, payable to IRE. Of this amount, $4.00 represents the PGEC fee, and $4.50 the affiliate membership.
2) Send your check and a letter requesting PGEC membership to

The Institute of Radio Engineers
1 East 79 St.
New York 21, N. Y.

No application blanks are needed.

Or

Contact your local IRE Section or PGEC Chapter directly.

Urge your friends who qualify to join the PGEC now.

Index to

IRE TRANSACTIONS

ON

ELECTRONIC COMPUTERS

Volume EC-10, 1961

IRE Transactions on Electronic Computers

Index to Volume EC-10, 1961

Contents

Volume EC-10, Number 3, September, 1961

Volume EC-10, Number 4, December, 1961

Author Index

Subject Index

INFORMATION FOR AUTHORS

IRE TRANSACTIONS ON ELECTRONIC COMPUTERS is published quarterly, in March, June, September, and December, with a distribution of over 9000 copies, largely to engineers, logicians, and supervisors in the computer field. Its scope includes: a) all aspects of design, theory, and practice relating to systems for digital and analog computation and information processing; b) components and circuits for digital and analog systems, including techniques for accomplishing the functions of logic, arithmetic, storage, control, mass data storage, input, output, and external communication in such systems; c) relevant portions of supporting disciplines, including switching theory, symbolic logic, numerical methods, codes and number representation systems, abstract machine or automation theory, bio-sciences, machine learning, pattern recognition, and other extensions of logical machine capabilities; d) production, testing, operation, and reliability of digital and analog systems; and e) those aspects of application, use, and programming of digital and analog computing devices and information processing systems that relate to their design and operation.

If a paper of widespread interest beyond the computer field is submitted, it will be recommended to the Editor of PROCEEDINGS OF THE IRE for publication. If our reviewers feel that a paper should be submitted to a different IRE TRANSACTIONS, we will so recommend to the author.

Publication time in IRE TRANSACTIONS ON ELECTRONIC COMPUTERS, from receipt of the original manuscript to mailing of the issue, is normally in excess of 5 months, but can be made as little as 3½ months if the occasion demands and the manuscript is carefully prepared.

To avoid delay, please be guided by the following suggestions:

A. Process for Submission of a Technical Paper

1) Send to the appropriate Editor three copies of your manuscript, each copy complete with illustrations. (For Letters to the Editor, two copies will do.)

2) Enclose originals for the illustrations, in the style described below. Alternatively, be ready to send the originals immediately upon acceptance of the paper.

3) Enclose a separate sheet giving your preferred address for correspondence and return of proofs.

4) Enclose a technical biography and photograph of each author, or be ready to supply these upon acceptance of the paper. For biography style, see any IRE journal.

5) If the manuscript has been presented, published, or submitted for publication elsewhere, please so inform the Editor. Our primary objective is to publish technical material not available elsewhere, but on occasion we publish papers of unusual merit that have appeared or will appear before other audiences.

B. Style for Manuscript

1) Typewrite, double or 1½ space; use one side of sheet only. (Good office-duplicated copies are acceptable.)

2) Provide an informative 100- to 250-word summary (abstract) at the head of the manuscript. It will appear with the paper and also separately in PROCEEDINGS OF THE IRE.

3) Provide a separate double-spaced sheet listing all footnotes, beginning with "*Received by the PGEC _____," and "†(Affiliation of author)," and continuing with numbered references. Acknowledgment of financial support is often placed at the end of the asterisk footnote.

4) References may appear as numbered footnotes, or in a separate bibliography at the end of the paper, with items referred to by numerals in square brackets, *e.g.*, [12]. In either case, references should be complete, and in IRE style.

Style for papers: Author (with initials first), title, journal title, volume number, inclusive page numbers; month, year.

Style for books: Author, title, publisher, location, year; page or chapter numbers (if desired).

See this or previous issues for further examples.

5) Provide a separate sheet listing all figure captions, in proper style for the typesetter, *e.g.*: "Fig. 1—Example of a disjoint and distraught manifold."

C. Style for Illustrations

1) Originals for illustrations should be sharp, noise-free, and of good contrast. We regret that we cannot provide drafting or art service.

2) Line drawings should be in India ink on drafting cloth, paper, or board. Use 8½×11 inch size sheets if possible, to simplify handling of the manuscript.

3) On graphs, show only the coordinate axes, or at most the major grid lines, to avoid a dense, hard-to-read result.

4) All lettering should be large enough to permit legible reduction of the figure to column width, perhaps as much as 4:1.

5) Photographs should be glossy prints, of good contrast and gradation, and any reasonable size.

6) Number each original on the back, or at the bottom of the front.

7) Note item B-5 above. Captions lettered on figures will be blocked out in reproduction, in favor of typeset captions.

Mail analog and hybrid computer manuscripts to:
John E. Sherman
Associate Editor, IRETEC
Lockheed MSD
Sunnyvale, Calif.

Mail logic and switching theory manuscripts to:
Prof. E. J. McCluskey
Associate Editor, IRETEC
Dept. of Electrical Engineering
Princeton University
Princeton, N. J.

Mail all other manuscripts to:
Prof. Norman R. Scott
Editor-in-Chief, IRETEC
Dept. of Electrical Engineering
University of Michigan
Ann Arbor, Mich.

D IN STACKS

IRE TRANSACTIONS ON ELECTRONIC COMPUTERS

Volume EC-10 **DECEMBER, 1961** **Number 4**

Reviews of Books and Papers in the Computer Field

(See front cover for main contents of issue)